DOCUMENT RESUME

ED 207 854 SE 035 734

AUTHOR Rissland, Edwina L.; Waisbrot, Sally
TITLE A Conceptual Framework for High School Mathematics. Final Report.
INSTITUTION Massachusetts Univ., Amherst. Dept. of Computer and Information Science.
SPONS AGENCY National Inst. of Education (DHEW), Washington, D.C.
PUB DATE Jul 81
GRANT NIE-G-80-0096
NOTE 183p.; Not available in hard copy due to marginal legibility of original document.

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Cognitive Objectives; Cognitive Processes; *Curriculum Design; Educational Objectives; Educational Research; Learning Theories; *Mathematical Concepts; Mathematics Education; Mathematics Instruction; Secondary Education; *Secondary School Mathematics; *Teaching Methods; Teaching Skills
IDENTIFIERS *Conceptual Frameworks; *Mathematics Education Research

ABSTRACT

Presented is the final report of a one-year project on the structure of high school mathematics knowledge sponsored by the National Institute of Education (NIE). The main body contains conclusions on a conceptual framework for representing high school mathematics. Results are presented through discussion of three domains from the secondary school curriculum: conic sections, quadrilaterals, and area. The main purpose of the project was to develop a conceptual framework to make it easier for teachers to describe, lay out, and present mathematics in a way that captures their knowledge of it. In addition to the main body of this document, there are three appendices: (1) Frames for Conics; (2) The Structure of Knowledge in Complex Domains; and (3) Teaching Learning Systems. (MP)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED207854

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

SE

A CONCEPTUAL FRAMEWORK FOR HIGH SCHOOL MATHEMATICS

FINAL REPORT - NIE-G-80-0096

EDWINA L. RISSLAND
SALLY WAISBROT

JULY 1981

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE
UNIVERSITY OF MASSACHUSETTS
AMHERST, MA 01003

SE 035 734

A Conceptual Framework for High School Mathematics

FINAL REPORT FOR GRANT NIE-G-80-0096

Edwina L. Rissland, Principal Investigator
Sally Waisbrot, Research Assistant

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

July 1981

TABLE OF CONTENTS

Summary of Activities.....1

1. Introduction.....3
2. An example from conics.....13
3. The second domain: quadrilaterals.....21
4. The third domain: area.....25

References.....29

Appendix 1: Frames for Conics
Appendix 2: The Structure of Knowledge in Complex Domains
Appendix 3: Teaching Learning Systems

SUMMARY OF PROJECT ACTIVITIES

This is the final report of a one-year project on the structure of high school mathematics knowledge sponsored by NIE (NIE-G-80-0096). The main body of this report contains our conclusions on a conceptual framework for representing high school mathematics; we present our results through discussion of three domains from the high school curriculum: conic sections, quadrilaterals, and area. In addition there are three appendices: Appendix 1 contains a set of frames for the conic section domain; Appendix 2 contains the paper "The Structure of Knowledge in Complex Domains" by the Principal Investigator; Appendix 3 contains the masters project report, "Teaching Learning Systems", by the Research Assistant.

This grant ran from 6/22/80 to 7/21/81. During the summer of 1980, the PI and the Research Assistant worked fulltime on this project. They perused several high school level mathematics texts and books and discussed specific subject areas and representation issues. Selected sections of text were read and mapped out in detail with particular attention to issues in structure and representation. The research assistant continued fulltime with this work for the 1980-81 academic year; the PI spent one fulltime week on this project in June 1981 wrapping things up.

In addition to activities carried out on campus in Amherst, the PI participated in two major off-campus activities: (1) the Fourth International Congress on Mathematics Education, held at Berkeley during August 1980; and (2) the NIE-LRDC Conference on Thinking and Learning Skills held in Pittsburgh during October 1980.

At ICME, the PI delivered a talk on the influences of artificial intelligence and cognitive science research on mathematics education. In addition, the PI attended the Fourth International Conference for the Psychology of Mathematics Education, also held at Berkeley, and the first annual meeting of the American Association for Artificial Intelligence, held at Stanford; at the AAAI she presented work from another research project. At the NIE-LRDC conference, the PI presented the paper entitled "The Structure of Knowledge in Complex Domains", which will appear in the proceedings of the conference to be published by Lawrence Erlbaum Associates. This paper well reflects the concerns and results of this grant.

A CONCEPTUAL FRAMEWORK FOR HIGH SCHOOL MATHEMATICS

1. INTRODUCTION

Goal

To improve the teaching and learning of mathematics we must better understand the structure of mathematics itself and the cognitive processes involved in understanding that structure [Papert, Davis]. We need to know what we know when we understand a body of mathematics, how we acquire such knowledge, and how we use it. These concerns are important to us both as teachers and as learners.

As teachers we must also know how to organize our knowledge and understanding of mathematics so that we can teach it to our students. In our presentations we must make clear such things as the main points, the threads of reasoning and the interconnections. It is imperative that we make clear such "global" or "meta-level" knowledge to our students since this sort of knowledge is apparent only after one has been through the subject, made sense of it, and related it to other topics; thus it is nearly impossible for students to acquire it while learning the subject for the first time. It is this overall understanding of the subject which we as teachers are in an excellent position to give to our students and which they will most likely fail to acquire without us.

The importance of this global sort of understanding is described by Polya and Szego [Polya and Szego]:

> "One should try to understand everything: isolated facts by collating them with related facts, the newly discovered through its connection with the already assimilated, the unfamiliar by analogy with the accustomed, special results through generalization, general results by means of suitable specialization, complex situations by dissecting them into their constituent parts, and details by comprehending them within a total picture".

Polya and Szego are asking us to see the forest among the trees. This is always difficult; especially so for students, who find the trees novel and who have no sense of the forest's extent. But this is precisely where we as teachers can help, for we know not only the botany of the trees, the shape and extent of the forest, but also how the forest connects with other forests and the general lay of the mathematical land.

The purpose of this project is to develop a conceptual framework to make it easier for teachers to describe, lay out and present mathematics in a way that captures their powerful overall knowledge of it.

There are often many ways to represent the same knowledge. The method chosen depends on our purpose: what we are going to use such knowledge for. Researchers in logic and theorem proving often use the predicate calculus to encode knowledge. The PI in her article "Understanding Understanding Mathematcs" created a conceptual framework for mathematical knowledge at the college level to investigate questions about learning, teaching and understanding.

The domain of high school mathematics requires that this representation be modified in several ways. High school mathematics emphasizes teaching and learning by questioning and discovery. Connections must be made between new concepts and their predecessors that make use of motivational material and exercises. For high school students, it is essential that applications link math with the real world and that the subject be weighted more toward concrete principles than abstractions.

We chose three areas of high school mathematics to represent: conic sections from algebra, quadrilaterals from geometry, and area, a topic which draws from both algebra and geometry. In the first, conic sections, we worked on the development of the most appropriate frame structure. The major part of each frame is a distillation of the information from the standard texts, Algebra II Structure And Method by Dolciani et al and Holt Algebra II by Nichols et al. Books such as In Mathematical Circles by Eves and Mathematics A Human Endeavor by Jacobs provide the real world applications, concrete examples, and other motivational material the standard texts lack.

In the second domain, quadrilaterals, we tried out the appropriateness of our frame and made necessary modifications. The final domain, area, was encoded using the modified frames as a test of our ideas on structure and representation.

2. THE FIRST DOMAIN: CONIC SECTIONS

High School Content

The text Algebra II examines the subject of conic sections in two isolated ways. First, during the study of quadratic equations, the parabola is introduced as the graphic representation of a quadratic equation and a useful tool for solving min-max problems. Later the conics are introduced as a group, defined in terms of a locus of points. Primarily, the relationship between equation and graph is emphasized. These graphs are also used as tools in solving inequalities and systems of equations. In a student's fourth year of high school math, the study often includes translation and rotation.

Spaces

We divided the domain into the five areas:

Concepts-space
Procedures-space
Results-space
Examples-space
Exercises-space.

These spaces were tailored to reflect the structure of high school mathematics.

Concepts-space showed an abundance of definitions. High school math assumes very little previous experience with conic sections and so each topic adds many new words. This formed the major part of Concepts-space. There were occasional mega-principles, which are heuristic capsulizations or words of wisdom, and counter-principles, which are words of warning.

High school students often need a link between the information they learn and how to apply it to exercises. Procedures-space frames were made up of leading questions or a series of increasingly difficult problems designed to encourage the discovery of a method of problem solving. For example, a procedural frame on equations of hyperbolas helps students uncover information they already know about graphing hyperbolas by asking:

1. How do you know this is the equation of a hyperbola?
2. What are the vertices?
3. What are the asymptotes?
4. What are the foci?
5. How does this information help you graph the hyperbola?

Proof is a less important concept in high school than it is in college mathematics. Consequently, the presentation of proofs is less formal and students are expected to be able to follow a demonstration of a proof rather than create it or reproduce it on their own. To account for this difference, Results-space is

itself less formal. A problem this creates is that the informal proof of Results-space resembles the procedure for use in Procedures-space making them at times hard to distinguish. A frame of Results-space might "prove" that all quadratic equations of the form ax**2 +bx+c=y can be represented as parabolas while a frame of Procedures-space would explain how to use the general quadratic form to graph parabolas.

The desire for informality which limited Results-space led to an explosion in Examples-space. This is easily explained. An alternative to a formal proof is a series of numerical examples which lead to a general conclusion. Sometimes these "proofs by induction" were supplemented by a more formal counterpart in Results-space, but just as often they stood alone since having too many proofs leaves the bulk of high school students confused and unsatisfied, and numerical examples are more convincing. The Holt text was interesting in that it used Examples-space to the exclusion of Results-space. All new information was introduced through numerical examples. Examples-space supported Procedures-space by providing model examples to use as the basis of procedures.

Exercises are an integral part of high school mathematics. Exercises in the texts were almost entirely those which practice a skill. For example, following an example in the text on graphing a hyperbola, an exercise asks the same thing. Having been taught a procedure for solving min-mix problems, a student is asked to find the maximun height reached by a tossed ball. Thus, exercises were tightly linked to a specific example or procedure frame.

Examination of the graphs of each space individually (figs. 1-5) shows the disproportionate sizes of the five spaces. But what may not have been apparent from the preceding discussion is the great breadth and shallow depth of each graph. This is characteristic of high school mathematics (except, perhaps, geometry). The student is given a taste of a multitude of topics but no one area is pursued extensively. One wonders if perhaps students do not appreciate the surface features of a subject until they are far into its depths and so a shallow treatment of many topics doesn't allow time for true understanding.

Taxonomic divisions within each space

There are two different types of procedure frames. A discovery procedure encourages, for example, the discovery of the role of the variables and constants of the parabolic form y=a(x-h)**2+k. The need for tightening the link between equation and graph led to the creation of procedure for-use frames. In the study of ellipses, these frames showed both how to graph an ellipse from various forms of the equation and how to identify the equation, given limited information about the graph.

Basic results are the foundation of a new theory. "Any quadratic equation can be represented as a parabola" shows the use of learned information to lead the student to a new theory. Key

results provide a foundation for or illustration of elements of Examples-space. The derivation of the formula for an ellipse from the definition supported the example frame in which foci and focal radii were numerical. Transitional results are important in explicitly linking related frames of Results-space. Standard textbooks introduce y=a(x-h)**2+k and y=1/4m x**2 separately as the equation of a parabola. This creates confusion for students, who wonder which is correct, or which to use.
Transitional frames were created to meet those concerns.

Since examples were used more often than concepts or results to introduce a new topic, start-up examples were most numerous. We split the catagory into two parts: motivational start-up examples which used an activity, story, or experiment to introduce a topic and problem start-up examples which used a numerical or word problem example to introduce a topic. Model examples illustrated common processes and can to some extent be used as a template in problem solving. Model examples suggest generalized forms which can be used when searching for broad attributes. (x**2)/(a**2) - (y**2)/(b**2)=1 is a model example for hyperbolas. Reference examples are standard, commonly occuring, concrete cases, such as y = x**2.

It is interesting to note that in the standard texts we examined there is no mention of counter-examples or counter-principles. Counter-examples would not be difficult to create: (x**2)+4(y**2)=-16 is an example that not all equations in x**2 and y**2 are conic sections in RxR. (x**2)+2xy+4(y**2)-2x-5y-45=0 is an example that not all second order equations can be reduced to conic form by completing the square. This lack of attention to counter-examples, meant to prevent confusion, instead creates the illusion that math procedes on a steady course never hesitating or taking a wrong turn.

Drill exercises help a student practice a skill. Less frequently the texts used exercises for other purposes. Some exercises were motivational, exciting interest in a subject. Leading exercises took the student one step from what was explicitly taught, often providing a transition to the next frame, while challenging exercises required the student to make a significant conceptual leap. Cummulative exercises combined previously learned information with the new lesson. Test exercises were those whose breadth made them appropriate for a test. Since all but drill exercises were generally neglected by texts, a special effort was made to find more of the remaining types.

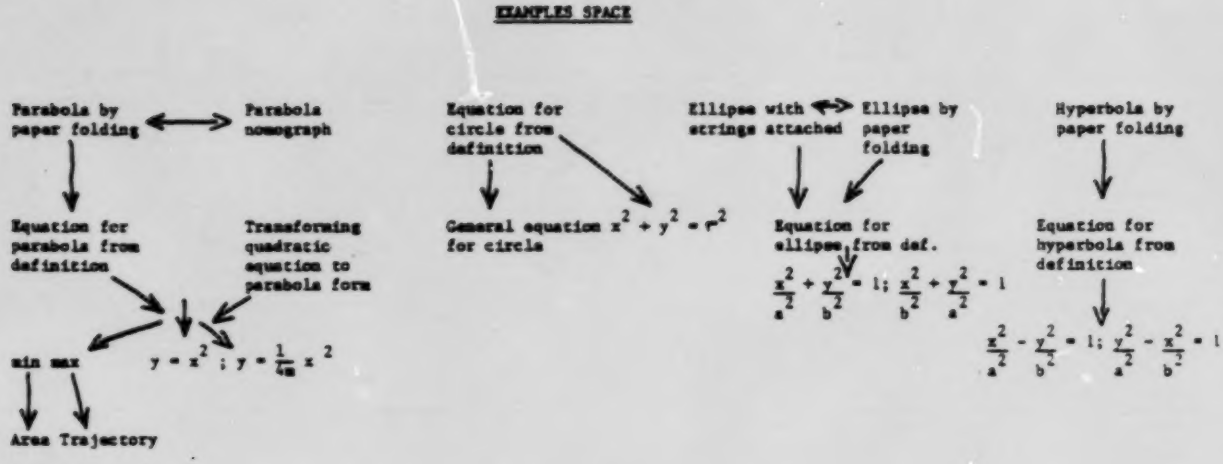

EXAMPLES SPACE
Parabola by paper folding
Parabola nomograph
Equation for parabola from definition
Transforming quadratic equation to parabola form
min max
$y = x^2$; $y = \frac{1}{4m} x^2$
Area Trajectory
Equation for circle from definition
General equation $x^2 + y^2 = r^2$ for circle
Ellipse with strings attached
Ellipse by paper folding
Equation for ellipse from def.
$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$; $\frac{x^2}{b^2} + \frac{y^2}{a^2} = 1$
Hyperbola by paper folding
Equation for hyperbola from definition
$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$; $\frac{y^2}{a^2} - \frac{x^2}{b^2} = 1$

Fig.1

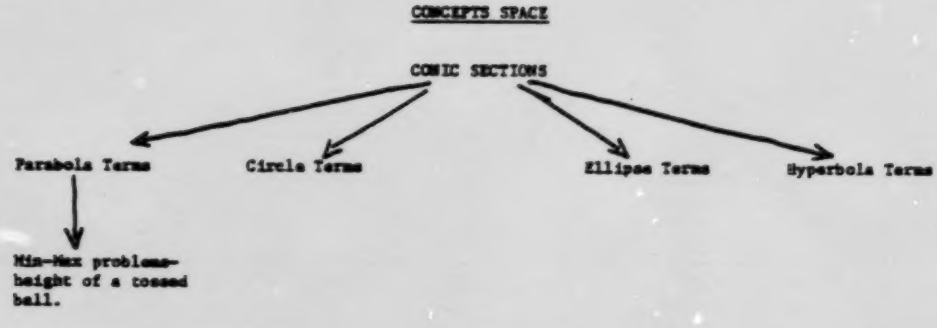


Fig. 2

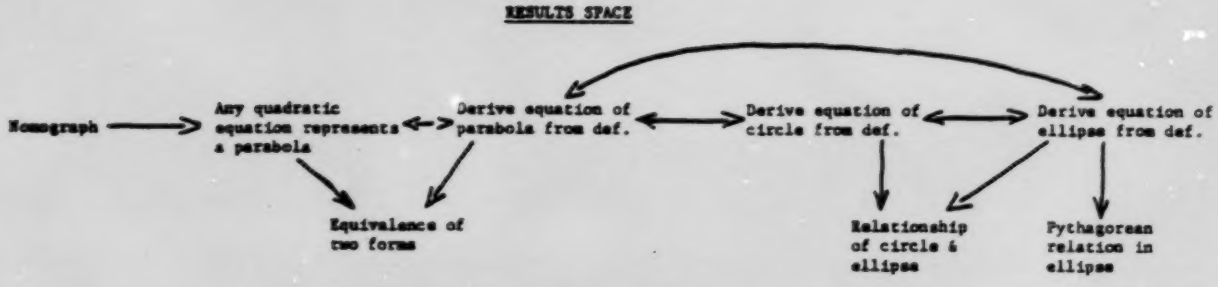

RESULTS SPACE
Nomograph
Any quadratic equation represents a parabola
Derive equation of parabola from def.
Derive equation of circle from def.
Derive equation of ellipse from def.
Equivalance of two forms
Relationship of circle & ellipse
Pythagorean relation in ellipse

Fig. 3

PROCEDURES SPACE

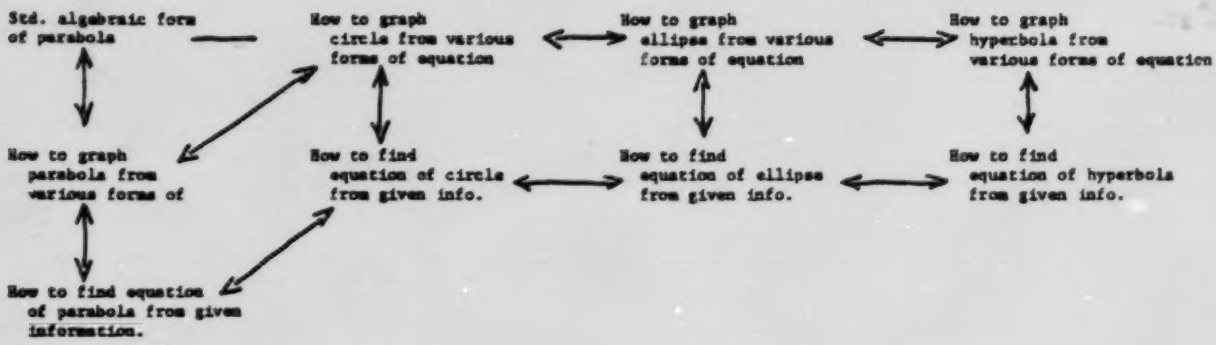

Std. algebraic form of parabola
How to graph circle from various forms of equation
How to graph ellipse from various forms of equation
How to graph hyperbola from various forms of equation
How to graph parabola from various forms of
How to find equation of circle from given info.
How to find equation of ellipse from given info.
How to find equation of hyperbola from given info.
How to find equation of parabola from given information.

Fig. 4

EXERCISES SPACE

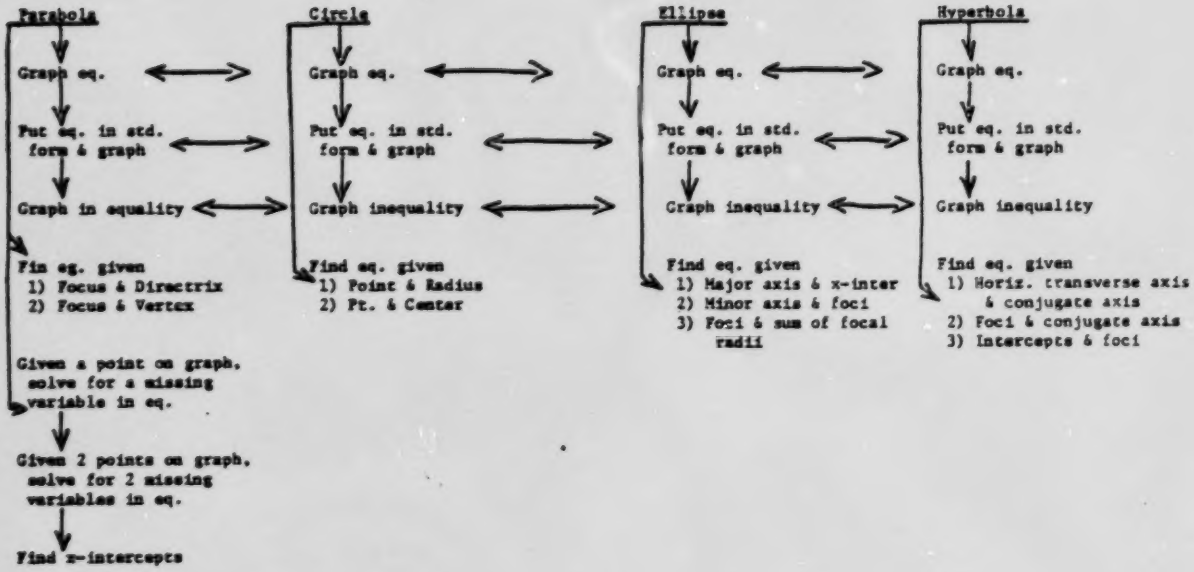

Parabola
Graph eq.
Put eq. in std. form & graph
Graph in equality
Fin eq. given
1) Focus & Directrix
2) Focus & Vertex
Given a point on graph, solve for a missing variable in eq.
Given 2 points on graph, solve for 2 missing variables in eq.
Find x-intercepts
Circle
Graph eq.
Put eq. in std. form & graph
Graph inequality
Find eq. given
1) Point & Radius
2) Pt. & Center
Ellipse
Graph eq.
Put eq. in std. form & graph
Graph inequality
Find eq. given
1) Major axis & x-inter
2) Minor axis & foci
3) Foci & sum of focal radii
Hyperbola
Graph eq.
Put eq. in std. form & graph
Graph inequality
Find eq. given
1) Horiz. transverse axis & conjugate axis
2) Foci & conjugate axis
3) Intercepts & foci

Fig. 5

The representation frames

Basically, the same representation frame (fig. 6) was used for procedures, results, and examples. Concepts-space consisted simply of definitions. Exercises-space was made up of the exercises and a pointer to its dual procedure or example. The frames are descriptive and capture the key information about a procedure, result or example. Appendix 1 contains frames for the conic section domain. For instance, the fifth frame (ID = Pa-C) is representative of procedural frames, it is the standard algebraic form of a parabola. It is important that the setting, R**2, was explicit, since students at this point are ready to learn about complex numbers, and will then be able to go back and explore the validity of theorems in this new setting.

The frames have a pedagogical slant. A descriptive sentence (SENT) identifies the frames in terms of overall content and objectives. The demonstration (DEMON) section consists of a list of pedagogical questions designed to encourage students to create their own examples and explore their own hypotheses. After students discover that the 5 in y=x**2+5 moves the graph up 5 units, it is hoped that they ask themselves how to move the graph down or horizontally. More detailed leading questions are meant to be used when no progress is being made by the students. Pictures are valuable for this frame. In particular, showing the graphs of the equations overlayed emphasizes the relationship between the change in the equation and the change in the graph. Remarks are reminders for the teacher of areas students find difficult, suggested hints, creative problems. Practical applications and integration suggest independent projects, experiments special "Friday" lessons.

The frames are interconnected. Inspace (FORWARD, BACK) and dual space pointers remind one of related frames: inspace pointers to frames within the same space, and dual-space pointers to strongly related frames from other spaces. ("C" indicates dual concepts, "E", dual examples, etc.) The foward pointer indicates that each of the conic sections will be introduced in a similar manner. The backward pointer helps one know what to review when difficulties are encountered.

ID CLASS NAME

STMT SETTING
SENT

DEMON

PICTURE

MATERIALS
REMARKS

PEDAGOGY
INSPACE FORWARD

BACKWARDS

DUAL

APPLICATION/INTEGRATION

Fig. 6 - The representation frame.

2. AN EXAMPLE FROM CONICS

In order to illustrate how the conceptual framework is formed, we outline the parabola portion of the conics domain. The text Algebra II and Trigonometry by Dolciani et al was used as the indicator of both the curricular content appropriate for high school and the order in which the material is presented. It forms the basis for most of the frames. However, a textbook is designed to meet the needs of a broad group of people. Every individual teacher and learner will find it lacking in some way. The learner may need to add frames for information the text assumed was learned previously. The teacher may wish to add more depth. We needed to add motivational frames to excite interest in a topic, transitional frames to link related topics explicitly, and procedural frames to help bind lesson and exercises more smoothly.

Parabolas are introduced as part of two different domains, polynomial functions and quadratic relations. In order to excite interest in this new topic, we added motivational frames for a parabolic nomograph and for creating a parabola by paper folding.

ID Pa-E CLASS Motivational NAME Parabola by paper folding
STMT SETTING R**2
SENT paper folding technique will illustrate def. of parabola, introduce terminology (focus, directrix, vertex, axis of sym., min or max point, envelope).
DEMON 1. Mark a point in the careful lower half of the center of a piece of paper.
2. Fold paper and make a sharp crease so careful lower edge falls along point.
3. Open paper flat and fold again at a different angle being careful lower edge again comes to marked point.
4. Repeat 20 times.

PICTURE

MATERIALS wax paper, flashlight
REMARKS 1. Wax paper is nice and projects on O.H.
2. Encourage dots in different places.
3. Make sure they fold from bottom edge only.
4. Ask whether size of paper changes result.
5. Challenge exercise- Book of Curves, p.7
PEDAGOGY Jacobs, H. A Human Endeavor, Chap. 6
INSPACE FORWARD Similar constructions of circle, ellipse, hyperbola.
BACKWARDS String drawing makes parabola envelope, similar triangles, reflections across line and about a point.
DUAL C- definition of parabola, symmetry, similarity
R- derivation of formula of parabola from def.
APPLICATION/INTEGRATION Flashlight and headlight beam, build solar parabolic oven

The text first defines f={(x,y): y=ax**2+bx+c, a,b,c,x in R, a=/=0} as a quadratic function and gives two example functions and their graphs: f(x)=x**2 and g(x)=x**2+3. The movement of the graph is noted as well as its symmetry and minimum point. Further examples show a maximum point, and movement of the graph along the horizontal axis. We combined these examples with questions into a discovery frame which leads students to discover the general form of a parabolic equation f(x)=a(x-h)**2+k, and critical information such as minimum or maximum, axis of symmetry, vertex, "width".

ID Pa-p1 CLASS discovery NAME standard algebraic form
STMT SETTING R**2
SENT Generate a general form of a quadratic equation which indicates vertex, axis of sym., "width", and orientation. Def'n: y=a(x-h)**2+k is the general form for a parabola.
DEMON 1. Have students graph y=x**2 by plotting points. Use O.H.
2. Now y=x**2+5. How does this change graph? Continue until pattern is apparent. Use overlay on O.H.
3. How can we change equation to move graph horizontally?
4. How can we turn it upside down?
5. How can we change width?
6. Ask for predictions about y=-2(x-4)**2+1.
7. How do all these changes affect axis of sym?
PICTURE

MATERIALS graph transparency, graph paper, overlay template of y=x**2
REMARKS caution: in y=x**2+5, 5 is not added to x but instead to x**2, so it affects the y value. Try with numbers to see. Show equation as y-5=x**2.

PEDAGOGY standard
INSPACE FORWARD same treatment with other conics

BACKWARDS general quadratic equation ax**2+bx+c=y

DUAL R- prove any quadratic equation can be represented graphically
as a parabola

Carefully chosen exercises are essential after each lesson to practice what was learned and to probe the student to think more deeply. Classroom exercises for this lesson drill the student in finding the critical information mentioned above. Further exercises fall into the catagories of drill and cummulative and end with several challenging exercises. A frame which indicates cummulative exercises acts as a warning flag to teacher and learner that review of old concepts may be necessary.

ID Pa-exer1 CLASS Cummulative NAME Parabolas in std form
STMT SETTING R**2
SENT Use the equation for parabola plus previously learned information to solve problems.
DEMON 1. The given ordered pair belongs to a quadratic fcn whose graph is symmetric with respect to the line with the given equation. Name another ordered pair that the fcn must therefore include a. (4,3); x=0 b. (-2,-1); x=-3
c. (m+p,q); x=m
2. One of the constants a,h, or k is undetermined. Find the value of the constant such that the graph of the given equation contains the given point.
a. y=a(x+3)**2+2; (2,-3) b. y=1/3 (x-h)**2-2; (-2,7)
3. Find the values of a and k such that the two given ordered pairs belong to the given fcn.
a. F: x-->a(x+4)**2+k; (-1,4), (2,31)
PICTURE

MATERIALS
REMARKS

PEDAGOGY Houghton Mifflin
INSPACE FORWARD

BACKWARDS Solving equations with 1 unknown; 2 unknowns.

DUAL C- symmetry-What it means for a graph to have an axis of symmetry.

APPLICATION/INTEGRATION

The text next develops the basic result that any quadratic equation can be represented graphically as a parabola. This is developed informally, via the example y=3x**2+6x+1, rather than through a formal proof. We included this as a model example since it serves as a template, and more theoretically as a basic result frame.

ID Pa-E* CLASS Model NAME Change a general quadratic equation to std parabola form.

STMT SETTING R**2

SENT Graphing a general quadratic equation will show it to be a parabola and motivate change in form of equation to make easier to find the critical points.

DEMON y=2x**2-12x+7

1. Plot points
2. Name vertex, axis of symmetry, min or max, width.
3. What would be usual parabola form for graph with these critical points?
4. Are the two forms of the equation equivalent?
5. How can we get from one to the other?
6. Continue number examples until class is ready for theoretical

PICTURE

MATERIALS graph paper

REMARKS might want to use easier exercise withot leading coefficient, e.g., y=x**2-4x+6

PEDAGOGY Standard

INSPACE FORWARD manipulating general equations for ellipse, hyperbola

BACKWARDS Completing square to discover quadratic formula

DUAL R- Proof for general case

APPLICATION/INTEGRATION

ID P2-R* CLASS Basic NAME any quadratic equation can be represented graphically as a parabola

STMT SETTING R**2

SENT Use the general equation for a quadratic ax**2+bx+c=y and show it to be a parabola.

DEMON Proof-Use completion of square to get form y=a(x-h)**2+k

Pedag Ques:

1. How are start and goal alike, different?
2. How many times does x appear and to what power?
3. In the past, how have you collected x terms of 2nd and 1st power?
4. Review completion of square technique. Why use it?
5. What is the effect of c, a?
6. What is vertex, axis of symmetry?

PICTURE

MATERIALS

REMARKS Start with numerical example. Encourage students not to memorize formula for vertex and axis of symmetry

PEDAGOGY Standard

INSPACE FORWARD Solving quadratic systems, similar treatment with conics

BACKWARDS Completion of square to prove quadratic formula

DUAL C- Discuss uses of each form of the equation

E- Numerical example for this problem, min and max problems

APPLICATION/INTEGRATION Min and max problems, trajectory problems Physics- take strobe light pictures of ball's path.

Anticipating its exercises, the text gives two start-up examples, one for finding vertex and axis of symmetry from a general quadratic equation and one for solving min-max problems. Given the diversity of min-max problems, We added several example frames. We also felt that students rarely understand how a parabola is related to a min-max problem and so we included a mega-principle frame to illuminate that concept.

ID Pa-C CLASS Mega-principle NAME Min-max problems
STMT SETTING R**2
SENT Discover the role of the parabola in solving min-max problems
DEMON 1. Use ex.: find dimensions of a rectangle whose perimeter is 48, and whose area is as great as possible.
2. Using string, guess for a while.
3. Make a table showing "all" possible pairs.
4. Graph perimeter vs. area. Where is correct answer?
5. Why was graph a parabola?
6. Remind students that it was the fact that the problem demanded a quadratic equation that insured that graph was a parabola and a min or max did exist.
PICTURE

MATERIALS string 48" long for each student.
REMARKS

PEDAGOGY
INSPACE FORWARD Direct and inverse variation

BACKWARDS

DUAL E-For further illustration show problems where min or max don't exist.

APPLICATION/INTEGRATION

Much later in the text, parabolas are reintroduced as one of the quadratic relations, or conic sections. No where are these two treatments to parabolas related (except in a note to recall on p. 260); so we created transitional frames to link the two roles of parabolas.

ID Pa-R CLASS Transitional NAME Why two forms?
STMT SETTING R**2
SENT Students often are confused by double introduction to parabolas. Compare the information given by each.
DEMON 1. What is the history of the two forms:
y=a(x-h)**2+k-->gen quad eq; y=1/4m x**2-->definition.
2. Isn't this also a parabola: y=ax**2+bx+c?
3. What does each form tell about the parabola:
A-vertex, axis, width, orientation; B-focus, vertex, directrix, width, axis, orientation; C-roots (where graph crosses axis.)
4. Go through finding these parts of A and B using numerical examples when necessary.
5. What is vertex of y=1/4m x**2? y=1/4m(x+3)**2+5?
6. Seeing this form, can you find focus and directrix in form A?

PICTURE

MATERIALS
REMARKS 1. Can start class with To Tell The Truth style "Will the real parabola please stand up."
Debate 3 cases.
2. In B, make sure they can find focus and directrix if vertex is changed.
PEDAGOGY
INSPACE FORWARD

BACKWARDS

DUAL

APPLICATION/INTEGRATION

This time the general form for a parabolic equation is derived from the definition: "the set consisting of every point P whose distance from a fixed point called the focus, is equal to the perpendicular distance from P to a line, called the directrix, that does not contain the focus". The axis of symmetry and vertex are redefined. Also a megaprinciple is stated that "the vertex is the midpoint of the segment of the axis between the focus and directrix." Again it was appropriate to derive this through an example, and then more theoretically. Exercises were drill oriented, graphing parabolas from equations and developing the equations, given focus and directrix. Many of the exercises could be considered model examples since exercises for the other conics mimic these.

3. THE SECOND DOMIAIN: QUADRILATERALS

To test the effectiveness of our representation frame it was important to experiment with it in diverse areas of high school math. Geometry with its emphasis on theorems and axiomatic proof seemed as far removed as possible from the procedural leaning of algebra. We were careful to choose a slice of geometry, namely quadrilaterals, which highlighted this emphasis, rather than mapping a unit on right triangles or circles, which would have been more closely linked to algebra.

The unit on quadrilaterals is made up almost entirely of theorems and their proofs. The subfigures in this group: quadrilateral, trapezoid, parallelogram, rectangle, square, rhombus, are introduced from the most general to the most specific since the theorems for the general quadrilateral are valid for the specific square and rhombus as well.

The information in three texts was combined for this mapping: Geometry (Houghton Mifflin), Geometry (H. Jacobs), and Geometry in Modules (M. Lange).

Spaces and divisions within each space

Concepts-space had a large number of definitions since each figure and its conponent parts (e.g., trapezoid, bases, legs, median) had to be defined. In addition, such an abundance of information was presented in a few short pages that the authors provided mega-principles to help the student keep track of important information. For example:

> "Remember, sides, angles, and diagonals will be important things to watch; sides will be equal or parallel, angles will be equal, right, or supplementary, and diagonals will be equal, bisect each other or bisect angles"
>
> "To prove a quadrilateral to be a parallelogram, try these five ways...".

Counter-principles offered helpful reminders such as:

> "It's possible to draw a quadrilateral with equal diagonals which is not a rectangle."
>
> "If one diagonal bisects the opposite angles, the quadrilateral does not have to be a rhombus".

Procedures-space was nearly empty in this domain. Perhaps the theorems and proofs of Results-space provide the needed procedural flow. On the other hand, a procedure may be an algebraic process which would be more apparent in analytic or coordinate geometry.

Results-space boomed with the theorem-proof-theorem-proof format of most plane geometry texts. The text Geometry in Modules was

more informal in its approach and consequently theorems were "validated" by experimentation. In a year long study of geometry, it is necessary to limit the number of theorems presented. Consequently, all were important and considered key results. Postulates (which did not appear in this unit) could be considered basic results. Corollaries formed a catagory of their own. Transitional results were not needed since each theorem was linked strongly to its predecessor and successor. Another subdivision of results closely couple with corollaries, called implicit results could be used to record those unstated theorems such as "Opposite sides of a square are parallel" which is a result of a square being a parallelogram.

Examples-space showed an interesting variety. Of course, each different quadrilateral had a model example. (Rhombus had two, one for its horizontal and one for its diamond orientation.) A chart matching specific quadrilaterals with their properties was a useful reference example. Other reference examples included a rhombus with its properties explicitly labelled and a group of parallelogram drawings which make obvious the fact that diagonals are not equal. A counterexample showed figures which aren't quadrilaterals. Motivational start-up examples were, for the most part, activity oriented as in dividing a card into seven equal parts using only a pencil and sheet of lined paper.

In the quadrilateral domain, exercises either asked for recall of information from a theorem or asked the student to generate a proof based on the theorems of the lesson. The former were drill exercises, for example, if ABCD is a parallelogram and AB=6 then CD=?. The latter was catagorized as cummulative since each proof was built on previous as well as current information. A leading exercise might ask the learner to investigate the properties of diagonals of the various quadrilaterals.

The representation frames

The same representation frame was used for all five spaces. The pedagogical slant is again achieved through leading questions. Also, presenting proofs with their reason column blank allows the learner a chance to make the transition between understanding a proof and generating one.

An example-Trapezoids

The subject of trapezoids will illustrate the representation scheme in the geometry domain. Subject matter from the three texts will be interwoven resulting in a balance of motivational, pure axiomatic, and inductive material. Most geometry texts split the quadrilateral domain into subsections based on different geometric figures. Lange, instead, divides the subject based on properties of sides, angles, and diagonals. Though we found this approach interesting we felt the conventional method had more appeal as a natural division and so we rearranged Lange's

material.

Jacobs begins his study of trapezoids with the following motivational example:

> "Can you figure out a simple way, using just an ordinary sheet of ruled paper, to divide one of the edges of a 3X5 card into seven equal parts? One could estimate the number, divide it by seven, and then try to use the result to mark the points on the card. Can you think of a simpler way? (Place the card at a slant so that two consecutive corners fall on lines that are seven spaces apart. Mark the points on the card at which the lines in between these two lines intersect the edge of the card.)"

After proving why this works, students are told that all the figures they have drawn are trapezoids.

All the texts then introduced the definitions of trapezoid, legs, bases, median, isosceles trapezoid. The learner would want to record a model example of the trapezoid with its parts labelled and parallel sides marked. The Houghton Mifflin text also recommends a reference example showing less common forms of the trapezoid: one right angle, bases as vertical segments, legs sloping the same direction.

The first result introduced is "Base angles of an isosceles trapezoid are equal". Strong backpointers link isosceles trapezoids to isosceles triangles and help learners predict theorems. The texts consider the choice of auxiliary lines from vertices perpendicular to opposite base too difficult for students to make, and so this and several other steps of the proof are given as hints. This result, however, has a pointer to the exercises where the same theorem is again to be proved, this time with one auxiliary line parallel to a leg. A corollary to this theorem is "The diagonals of an isosceles trapezoid are equal."

Lange next suggests an exploration of length of the median of the trapezoid in relation to its bases and presents the result:

> The median of a trapezoid has two properties: 1) It is parallel to the bases and 2) Its length equals half the sum of the lengths of the bases.

Again since the proof of this is quite difficult the authors suggest presenting the steps and allowing students to supply reasons. Houghton Mifflin next puts forth an interesting discovery procedure. A series of trapezoids are shown with medians drawn. Each figure has one of the bases grow shorter and shorter until this short base becomes merely a point, i.e., the figure is now a triangle. Students are thus led to discover that the segment that joins the midpoint of two sides of a triangle 1) is parallel to the third side and 2) half as long as the third side.

The unit ends with a large number of exercises, probably because the trapezoid is the least familiar quadrilateral for most. Drill exercises which can be done mentally include marking the measure of missing sides and angles and medians. A more cummulative example asks the same questions using variables for the sides and angles. A drill exercise which tests understanding of definition asks the learner to draw a trapezoid if possible, having certain characteristics such as three equal sides. Challenging exercises ask if the diagonals of a trapezoid can bisect each other or ask what figure is formed when consecutive midpoints of an isosceles trapezoid are joined. Proofs test whether students can apply the new theorems in conjunction with previously learned information about triangles.

4. THE THIRD DOMAIN: AREA

Having created and tested our representation frame for the domains of conics and quadrilaterals respectively, we chose to explore a third domain while holding the frame fixed. We chose area, a topic sharing the flavor of both algebra and geometry. As before, we found that the five spaces were unequally represented. This time, examples and exercises spaces were the largest. Our representation frame, however proved flexible enough to handle the variations in emphasis.

Representation Spaces

Units on area most commonly appear in geometry texts. We again integrated the texts Geometry published by Houghton Mifflin, Geometry by Harold Jacobs, and Geometry in Modules by Muriel Lange. The high school treatment of this topic encompasses a bare bones introduction to a specific formula, possibly with its derivation, followed by numerous exercises requiring creative applications of the formula. Lange begins with an exploration of the concept of area and units of measurement. All the texts progress from the simplest square or rectangle to general polygons and circles.

Concepts-space as before, consisted mainly of definitions. Area and region were defined as well as the parts of figures needed in the area formula. Lange offers quite a few mega-principles in developing the concept of area:

> "Circles make a bad standard unit of measurement"
>
> "To find the area of a holey shape, subtract the areas of the smaller regions from the whole".

In the quadrilateral unit we hypothesized that the void in Procedures-space was due to the procedural flow of information in the proofs of Results-space. Having fewer proofs, the area domain showed an increase in procedures. Some themes were: finding the area of a trapezoid by dividing it in various ways; finding the area of an irregular object by averaging outer and inner areas, and discovering the relationship between area and perimeter. These procedures were discovery-oriented and emphasized exploration. This contrasts with the procedures for use of the conics domain.

Results-space contained all the basic area formulas with the addition of Heron's theorem and the formula for the area of an equilateral triangle. In the conics domain, informal proof was the rule. Quadrilaterals showed formal two-column proofs. Area, as expected, chose a middle ground. In general, a formal proof was described in paragraph form. Lange and Jacobs added results based on relationships:

Triangles with equal bases and altitudes have equal areas.

If two triangles have equal bases, their areas are in the same ratio as their heights;. Of all rectangles with a given area, the one with the smallest perimeter is a square.

Examples-space and Exercises-space were large, containing numerous motivational examples and leading and challenging exercises. Area, being a subject of enormous practical use, provides one with large numbers of real world examples and applications.

Model examples showed polygonal regions divided into triangles for area computation, and sample rectangles, parallelograms, triangles, and trapezoids with bases and altitudes drawn. Start up examples demonstrated how to find area given unusual parts, or find a base, given area. A figure showing triangle removal to remind students of the derivation of the parallelogram formula is a reference example. Motivational examples include topics of land speculation, population density, tesselations.

As in the other two domains studied, area lessons were followed by numerous drill exercises to practice the technique of finding area. Although many exercises offered were beyond the drill level, the similarity of format from lesson to lesson caused many non-drill exercises to reach the stage of rote practice. For example, each lesson asked the student to find the area of a shaded region whose holes and irregularities corresponded to the lesson at hand, or given the area, find the height of a figure. Cummulative exercises used information about 30-60-90 and isosceles right triangles to find the area of triangles and trapezoids. Challenging exercises asked for a proof of the area formula of a parallelogram for the different cases, o asked why there is not a formula for the area of a general quadrilateral based on side length.

Representation frames

Our representation frame proved satisfactory for this domain. The slots of the frame were found to be necessary to cover adequately the subject and none were consistently left empty. The slot for applications and integration had been one we considered important to include in a high school course but it had been difficult to fill. Most texts do not expend much effort in this direction. In the area domain, however, this slot was always filled, since, as previously stated, area is a subject of much practical use. At times, this part of the frame was so important that it became the major emphasis of the frame.

An Example

To illustrate the use of the representation frames in the area domain, we will compare the treatment of circles in the Lange and Jacobs texts.

Jacobs begins with a motivational exercise:

> "Consider the earth as 25,000 miles long and having a tight ribbon around it. Now slit the ribbon and insert one foot. Pull the ribbon away so that it is equally far away from the earth at all places. How large is the gap?"

This reviews the definitions associated with a circle and introduces the relationship between circumference and radius. Lange, too, begins by adding circle definitions to Concepts-space.

Both authors introduce the area formula with a discovery procedure. Jacobs uses model examples to demonstrate that the perimeters of polygons inscribed in a circle get closer and closer to the circumference of the circle as the number of sides increases. A similar example shows the same process for area of a circle. He then introduces the concept of a limit with the mega-principles:

1. The limit of a constant is the constant itself.
2. The limit of a product is equal to the product of the limits.
3. lim v=pi where v=nsin(180/n).
4. lim cos(180/n)=1.

He reinforces these concepts with reference examples lim 2=2 and lim 2r=lim2*lim r. Next he proves area=pi r**2 by using his definition of area as a limit and using the limit facts.

Lange's model exercise shows a circle divided into segments. The demonstration portion of the discovery procedure is:

1. How many congruent wedges has the circular region been divided into? What kind of polygon does each wedge resemble?

2. Imagine unrolling the upper and lower semicircular regions and fitting them together to form a figure which resembles a parallelogram. What is the area formula for a parallelogram?

3. The radius of the circle is equal in length to the ? of the parallelogram. Half the circumference of the circle is equal to the ? of the parallelogram.

4. Use these measurements to find the area of this parallelogram. How does this compare to the area of the circle?

5. In getting the formula you assumed the wedge was a triangle and the figure formed was a parallelogram Why is this only approximately correct?

6. Would the approximation improve if there were 1000 wedges instead of 16? one billion?

Lange includes the mega-principle: To get exact area, leave the symbol pi in your answer. To get an approximate area, replace pi with 22/7 or 3.14.

Both authors have drill exercises in which the area of a circle is computed directly and computed as a part of another figure. Lange includes a number of challenging exercises which involve practical applications.

Lange continues with another discovery procedure in which the radius of a circle is doubled, tripled, etc. and students look for a corresponding relationship in areas. The conclusions, in Results-space is "If the radius of a circle is multiplied by n, its area is multiplied by n**2."

REFERENCES

Davis, R., Detailed Description of Mathematical Behaviors that Demonstrate Understanding. Proposal to joint NIE/NSF Program, 1979.

Davis, R. B., Jockusch, E., and C. McKnight. "Cognitive Processes in Learning Algebra". The Journal of Children's Mathematical Behavior, Vol. 2, No. 1, Curriculum Laboratory, University of Illinois at Urbana-Champaign, 1978.

Davis, R. B., and C. C. McKnight, The Conceptualization of Mathematics Learning As a Foundation of Improved Measurement. Development Report No. 4 (Final Report HEW NIE/G 76 0085), The Curriculum Laboratory, College of Education, University of Illinois, Urbana, 1979.

_____, "Modeling the Processes of Mathematical Thinking". The Journal of Children's Mathematical Behavior, Vol. 2, No. 2, Curriculum Laboratory, University of Illinois, 1979.

Dolciani, M. et al. Algebra II and Trigonometry. Houghten Mifflin, Boston, 1968.

Jacobs, H., Mathematics - A Human Endeavor. W. H. Freeman and Co., San Francisco, 1970.

_____, Geometry. W. H. Freeman and Co., San Francisco, 1970.

Lockwood, E. H., A Book of Curves. Cambridge University Press, Cambridge, 1961.

Minsky, M. L., "A Framework for Representing Knowledge" in [Winston].

Nichols, E. D. et al. Algebra II with Trigonometry. Holt, Rinehart, and Winston, New York, 1974.

Papert, S. A., Teaching Children to be Mathematicians Versus Teaching About Mathematics. MIT A. I. Memo No. 249, 1971.

_____, "Theory of Knowledge and Complexity". Lecture to IFIPS World Congress on Computers in Education, Amsterdam, The Netherlands, 1970.

[MD] Polya, G., Mathematical Discovery, Vol 2, Wiley, New York, 1962.

_____., How To Solve It. Second Edition, Princeton University Press, New Jersey, 1973.

_____, Induction and Analogy in Mathematics. Volume 1 of Mathematics and Plausible Reasoning. Princeton University Press, New Jersey, 1973.

Polya, G., and G. Szego, Problems and Theorems in Analysis, Vol 1. Springer-Verlag, New York, 1972.

Rissland (Michener), E. L., "Understanding Understanding Mathematics". Cognitive Science, Vol. 2, No. 4, 1978. Also published as A. I. Memo No. 488, MIT Artificial Intelligence Laboratory, 1978.

_____, The Structure of Mathematical Knowledge. Technical Report No. 472, MIT Artificial Intelligence Laboratory, 1978.

Rissland, E. L., Constrained Example Generation. COINS Technical Report, COINS, University of Massachusetts, Amherst, in preparation.

Soloway, E. M., and E. L. Rissland, Overview of an Example Generation System. Proceedings First National Conference on Artificial Intelligence, Stanford, August 1980.

Winston, P. H., The Psychology of Computer Vision. McGraw-Hill, New York, 1975.

APPENDIX 1

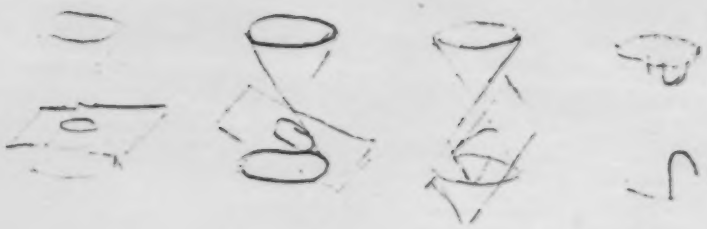
ID Conic-C CLASS Definition NAME Conic Sections

STMT SETTING R^2

SENT A conic section is a curve formed by the intersection of a plane and a cone.

DEMON
1. Tell story of Apollinus
2. Have students search for all possible shapes formed by slicing cone. Try to get them to describe how they formed slice so we can classify shapes as the same of different.
3. Add second nappe to help show difference between hyperbola and parabola
4. Talk about existance of degenerates (point,line, intersecting lines)
5. See if they can find how to slice cones for degenerates.

PICTURE



MATERIALS Clay, plastic knives, newspaper, wire might be better than knives

REMARKS Students will need "proof" that parabola and hyperbola are different.

PEDAGOGY Eves, H. In Mathematical Circles

INSPACE FORWARD Further def. of each conic in locus terms

BACK Previous experiments with polygons

DUAL

APPLICATION INTEGRATION History: Apollonius, Greek culture of his time; English, latin: ellipsis, hyperbole

ID Pa-R CLASS Transitional NAME Parabolic Nomograph Proof

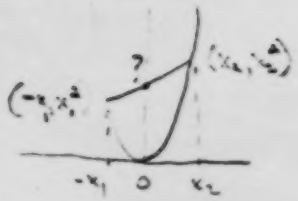
STMT SETTING R^2

SENT Use knowledge of linear equations to prove why parabolic nomograph works.

DEMON PROOF 1. Call two points to multiply, x1 and x2
2. Find equation of line through those points.
3. Solve for y-intercept by substituting point on curve and line.

PEDAG Q's 1. What points appear to be important
2. What are their coordinates?
3. can you see anything in the picture that you know nore about than parabolas?
4. What form of the equation of a line would be useful?
5. What missing values can you solve for?
6. What have you shown?

PICTURE



MATERIALS Large nomograph

REMARKS Make sure they have saved nomograph so they can see pattern of ordered pairs: $(1,1),(3,9),\ldots(x,x^2)$
This might be good as an extra credit hw problem, given 1 or 2 hints.

PEDAGOGY Jacobs, H. A Human Endeavor Ch. 6

INSPACE FORWARD coordinate geometry

BACK equation of line, graphing lines, y-intercept and slope

DUAL E nomograph exercise

APPLICATION Wall chart with movable bar for elementary school
INTEGRATION history: Decartes(joke-Eves)

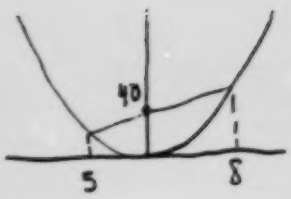
ID Pa-E CLASS Motivational NAME Parabola Computer(nomograph)

STMT SETTING R^2

SENT Show that connecting two points on the parabola $y=x^2$ produces the product of their x values at the y-intercept.

DEMON
1. Draw a pair of axes. Number the axes, letting 1" on the x-axis represent 10 and 1" on the y-axis represent 100.
2. Plot the points satisfying $y=x^2$ for $-30 \leq x \leq 30$
3. Join the points with a smooth curve
4. To multiply 5 and 8, find these numbers on the x-axis, one to the left and the other to the right of the origin.
5. Locate the two points on the curve directly above these points and use a straightedge to join them with a line.
6. The point where this line crosses the y-axis indicates the answer, 40.

PICTURE



MATERIALS Graph paper 10 sq/in. , straightedges, table of squares

REMARKS A table of squares would prevent carelass mistakes

PEDAGOGY Jacobs, H. A Human Endeavor Ch 6

INSPACE FORWARD

BACK Addition and multiplication nomographs

DUAL R Proof, compare slide rule nomograph and parabola nomograph

APPLICATION
INTEGRATION

ID Pa-E CLASS Motivational NAME Parabola by paper folding

STMT SETTING R^2

SENT Paper folding technique will illustrate def. of parabola and introduce terminology (focus, directrix, vertex, axis of sym., min or max point, envelope)

DEMON
1. Mark a point in the lower half of the center of a piece of paper
2. Fold paper and make a sharp crease so lower edge falls along point.
3. Open paper flat and fold again at a different angle being careful lower edge again comes to marked point.
4. Repeat 20 times

PICTURE

MATERIALS wax paper, flashlight

REMARKS
1. Wax paper is nice and projects of O.H.
2. Encourage dots in different places.
3. Make sure they fold from bottom edge only.
4. Ask whether size of paper changes result.
5. Challenge exercise-Book of curves, p.7

PEDAGOGY Jacobs, H. A Human Endeavor Ch. 6

INSPACE FORWARD Similar constructions of circle, ellipse, hyperbola

BACK String drawing makes parabola envelope, similar triangels, reflections across line and about a point.

DUAL C definition of parabola, symmetry, similarity

R Derivation of formula of parabola from def.

APPLICATION Flashlight and headlight beam,
INTEGRATION Build solar parabolic oven with science teacher

ID Pa-C CLASS ~~Procedural definition~~ NAME Standard algebraic form of a parabola

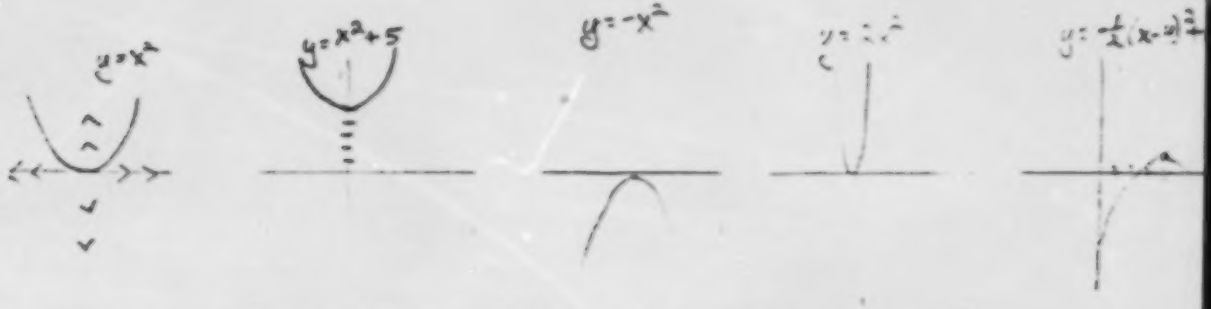
STMT SETTING R^2

SENT Generate a general form of a quadratic equation which indicates vertex, axis of symmetry, "shape", and orientation

DEF $Y=A(X-H)^2+K$ is the general form for a parabola

DEMON
1. Have students graph $y=x^2$ by plotting points. Show on O.H.
2. Now $y=x^2+5$. How does this change graph? Cont. until pattern is apparent. Use overlay on O.H.
3. How can we change equation to move graph right or left?
4. How can we turn it upside down?
5. How can we change width?
6. Ask for predictions about $y=-2(x-4)^2+1$.
7. How do all these changes affect axis of symmetry?

PICTURE



MATERIALS Graph trasparency, graph paper, overlay template of $y=x^2$

REMARKS Caution: in $y=x^2+5$, 5 is not added to x (but instead, x^2) so effects y value, not x. Try with numbers to see. Show equation as $y-5=x^2$

PEDAGOGY Standard

INSPACE FORWARD Same treatment of other conics

BACK general quadratic equation $ax^2+bx+c=y$

DUAL R prove any quadratic equation can be represented graphically as a parabola.

APPLICATION
INTEGRATION

ID Pa-R* CLASS Basic NAME Any quadratic equation can be represented graphically as a parabola

STMT SETTING R^2

SENT Use the general equation for a quadratic $ax^2+bx+c=y$ and show it to be a parabola.

DEMON PROOF Use completion of square to get form $y=a(x-h)^2+k$

PEDAG Q's
1. How are start and goal alike, different?
2. How many times does x appear and to what power?
3. In the past how have you collect x terms of 2nd and 1st power?
4. Review completion of square technique? Why should we use this?
5. What is the effect of c, a?
6. What is vertex, axis of symmetry?

PICTURE

MATERIALS

REMARKS Start with numerical example. Encourage students not to memorize formula for vertex and axis of symmetry

PEDAGOGY Standard

INSPACE FORWARD Solving quadratic systems, similar treatment of ci,el,hy

BACK Completion of square to prove quadratic formula

DUAL C Discuss uses of each form of the equation

E Numerical example for this problem, min and max problems

APPLICATION Min and max problems, trajectory problems
INTEGRATION Physics, take strobe light pictures of ball's path and graph

ID Pa-E* CLASS Model NAME Change a general quadratic equation to standard parabola form

STMT SETTING R

SENT Graphing a general quadratic equation ($ax^2+bx+c=y$) will show it to be a parabola and motivate change in form of equation to make it easier to find the critical points.

DEMON $y=2x^2-12x+7$
1. Plot points
2. Name vertex, axis of symmetry, min or max, shape.
3. What would be usual parabola form for graph with these critical points?
4. Are the two forms of the equation equivilent?
5. How can we get from one to the other
6. Continue number examples until class is ready for theoretical.

PICTURE

MATERIALS graph paper

REMARKS Might want to use easier exercise without leading coefficient, e.g. $y=x^2-4x+6$

PEDAGOGY Standard

INSPACE FORWARD manipulating general equations for ellipse, hyperbola

BACK Completing squmre to discover quadratic formula

DUAL R Proof for general case

APPLICATION
INTEGRATION

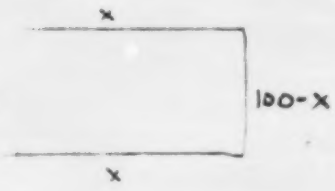
ID Pa-E¢ CLASS Start up NAME Min-max problems

STMT SETTING R^2

SENT The School is going to build an outdoor swimming lpool and the junior class donates a fence 200 m. long. What are the area and dimensions of the largest pool (rectangular) which could be enclosed by the fence?

DEMON 1. Draw a picture of pool.
2. Can any dimensions be labelled? Is there any relationship between the dimensions?
3. what is the area of the pool?
4. What kind of equation is this? What figure does it represent?
5. How does knowing this is a parabola help find the maximum area?
6. Put equation in parabola form. What is the maximum?
7. How can this ordered pair be interpretted in terms of the problem?
8. What if the problem had asked for minimum area?
9. If the rectangular restriction were removed, any ideas on the best answer?

PICTURE



$A = x(100-x)$

MATERIALS

REMARKS Watch out for students interpretting largest as >.

PEDAGOGY Standard

INSPACE FORWARD Trajectory problems

BACK Word problems with polynomial equations

DUAL C Procedural def of height a ball reaches

APPLICATION economics: optimal situation; linear programming
INTEGRATION

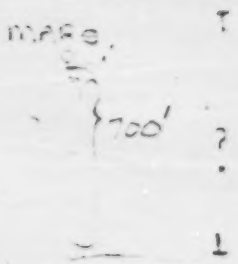
ID Pa-E∅ CLASS Start up NAME Min-max problem-trajectory

STMT SETTING R^2

SENT The gravitational constant on Mars is 5.6 ft/sec.^{2}. If I stand on the brink of a 700 ft. Martian cliff and throw a stone upward with an initial velocity of 112 ft./sec. what will be the maximum height it reaches?

DEMON
1. Follow procedure outlined on frame Pa-C∅.
2. If only gravity were at work what would equation be?
3. If we add the initial height of the ~~xxii~~ stone, how does that change things?
4. How does the initial velocity effect the equation? Why do they call it <u>initial</u> ~~xxi~~ velocity?

PICTURE



MATERIALS Ball

REMARKS When asking questions, do the experiments with a ball.

PEDAGOGY Standard

INSPACE FORWARD Solving general types of min-max problems

BACK Area problems

DUAL C Procedural def of height a ball reaches

APPLICATION Physics: strobe light experiments; take photos and graph results.
INTEGRATION

IDPa-C∅ CLASS Procedural def NAME Min-max problems:height a ball reaches

STMT SETTING R^2

SENT Generate a formula for height of a ball tossed upward:
$h=I+vt-gt^2$ (I=initial height, v=velocity, g=gravity, t=time)

DEMON
1. Throw a ball, ask how high it reached.
2. If I throw it again, will the answer be the same? What does the result depend on? (initial height of ball, velocity, gravity, time)
3. How does each variable element effect the whole? (I-constant, v-linear, g-quadratic, all vary with time)
4. Create formula by building, start with 0 initial height and velocity, then add height, and last, velocity.

PICTURE

MATERIALS Ball

REMARKS Could we solve problem on a different planet?

PEDAGOGY Standard

INSPACE FORWARD [illegible]

BACK Standard equation for a parabola

DUAL E #specific problems (Pa-E∅

APPLICATION Find out about gravity on other planets.
INTEGRATION

ID Pa-R # CLASS Key NAME Derivation of equation for parabola from definition

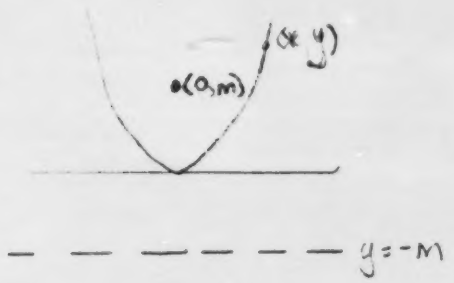
STMT SETTING R^2

SENT Using the definition-the locus of points equal distance from a fixed point and a line-form an equation.

DEMON PROOF 1. Focus(0,m), Directrix y=-m;
2. Dist F-(x,y)= $\sqrt{x^2+(y-m)^2}$
3. Dist (x,y)-line= $\sqrt{0+(y+m)^2}$
4. Eq= $x^2+(y-m)^2=(y+m)^2 \rightarrow x^2=4ym \rightarrow y=1/4m\ x^2$

PEDAG Q's 1. If we want to formulate equation for general case with vertex (0,0), how can we choose points? How should we name vertex, focus, directrix?
2. What does the definition say about the relationship between these parts?
3. How can we use previously learned information or problems to form an equation?

PICTURE



MATERIALS picture on OH

REMARKS 1. Do a number example first.
2. Point out that the above procedure is not for the most general case of the parabola.
3. Discuss what might be the most general case.

PEDAGOGY Standard

INSPACE FORWARD same for circle, ellipse, hyperbola

BACK $y=a(x-h)^2+k$ form of graph

DUAL startup
E example
C Procedure to convert general quadratic equation to parabola form
Dist between two points, between a point and line, focus, directrix

APPLICATION Hall of Whispers-capitol
INTEGRATION

ID Pa-E# CLASS Start up NAME Equation for parabola from definition

STMT SETTING R^2

SENT Choosing a focus and directrix, find the equation for the corresponding parabola.

DEMON
1. Have class choose point and line
2. Pick series of points equadistant from point and line. Does it look like a parabola? Will we get the parabola we're used to?
3. Label one point (x,y)
4. What does definition say about this point?
5. What is the distance to focus? to line? What is equation?

PICTURE

MATERIALS

REMARKS
1. Suggest directrix be vertical or horizontal on first try.
2. Even distance between focus and directrix makes things easier.
3. After a simple example is understood, encourage unusual selection of focus and directrix and predictions about results.
* 4. Challenge problem-Try abouve exercise with directrix y=x, focus (4,4)

PEDAGOGY Standard

INSPACE FORWARD same for other conics

BACK

DUAL C Writing good definitions

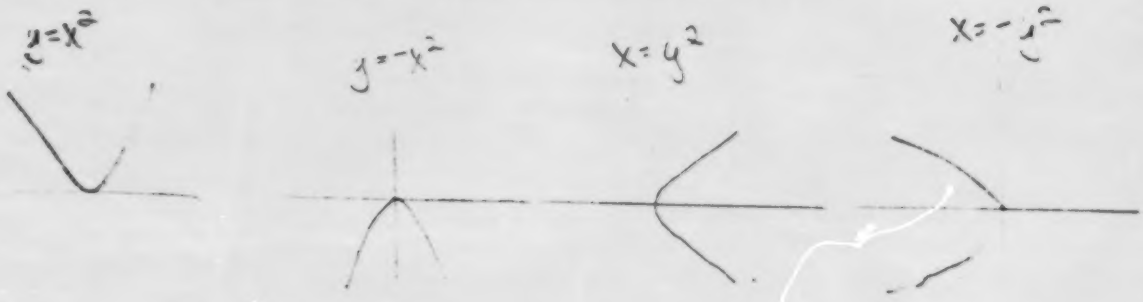
APPLICATION
INTEGRATION

ID Pa-E CLASS Reference NAME $y=x^2$; $y=1/4m\ x^2$

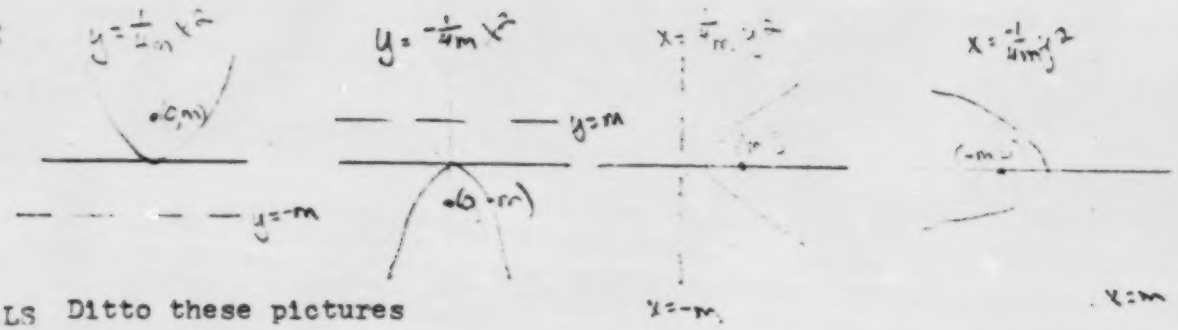
STMT SETTING R^2

SENT These basic forms of parabolas centered at (0,0) are the first to use when trying to answer general questions or solve general problems about parabolas.

DEMON 1. Ask how graph changes ~~xxx~~ as focus changes.



PICTURE



MATERIALS Ditto these pictures

REMARKS 1. These are a useful first case but are not the most general.
2. Useful to perturb.

PEDAGOGY Standard

INSPACE FORWARD circles, ellipses, hyperbola

BACK numerical examples

DUAL R proof of equation

APPLICATION
INTEGRATION

ID Pa-R CLASS Transitional NAME Why two forms: $y=a(x-h)^2+k$, $y=1/4m\ x^2$

STMT SETTING R^2

SENT Students often are confused by double introduction to parabolas. Compare the information given by each form.

DEMON
1. What is the history of the two forms A: Gen quad eq B Def
2. Isn't this also a parabola: C $y=ax^2+bx+c$
3. What does each form tell about parabola: A vertex, axis, shape, orientation B focus, vertex, directrix, shape, axis, orientation C roots (where graph crosses axis)
4. Go through finding these parts of A and B using numerical examples when necessary.
5. What is vertex of $y=1/4m\ x^2$? $y=1/4m(x+3)^2+5$?
6. Seeing this form, can you find focus and directrix in form A?

PICTURE

MATERIALS

REMARKS
1. Can start class with To Tell The Truth style "will the real parabola please stand up" Could have students debate for three cases.
2. In B, make sure they can find focus and directrix if vertex is changed.

PEDAGOGY

INSPACE FORWARD

BACK

DUAL

APPLICATION
INTEGRATION

ID Ci-R5 CLASS Key NAME Derivation of equation for circle from definition.

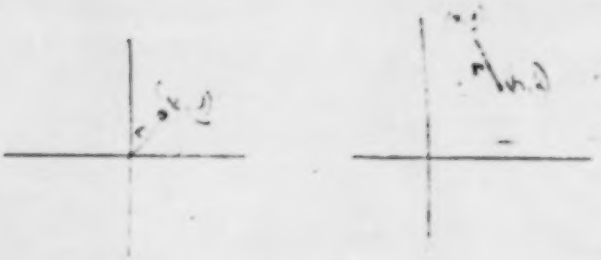
STMT SETTING R^2

SENT Using def-all points a given distance from a fixed point, form an equation for a circle.

DEMON PROOF 1. Center (h,k), radius r
2. Distance $(x,y)-(h,k)=\sqrt{(x-h)^2+(y-k)^2}=r$
3. $(x-h)^2+(y-k)^2=r^2$

PEDAG Q's 1. What would you name center to make it most general?
2. What would you name radius?
3. What would you name a point on the circle?
4. What is the distance between the center and the point on the circle?
5. What does the equation say

PICTURE



MATERIALS

REMARKS Use a numerical example to help.

PEDAGOGY Standard

INSPACE FORWARD ellipse, hyperbola

BACK parabola

DUAL E numerical example

c unit circle

APPLICATION
INTEGRATION

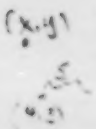
ID C1-E1 CLASS Start up NAME Equation of circle from definition

STMT SETTING R^2

SENT Choosing a center and radius, find corresponding equation for circle

DEMON 1. Pick a center and length of radius. Draw a picture following the definition.
2. Are there any other points you can label?
3. How does one label a general point?
4. What does the difinition tell about distances?
5. What is the equation for this circle?

PICTURE



MATERIALS

REMARKS 1. Parabola was easier because definitions stated two distances which were equal. Here students will find a distance and wonder what it is equal to.
2. Students will be able to find several points on the circle if radius is drawn horizontally

PEDAGOGY Standard

INSPACE FORWARD

BACK

DUAL

APPLICATION
INTEGRATION

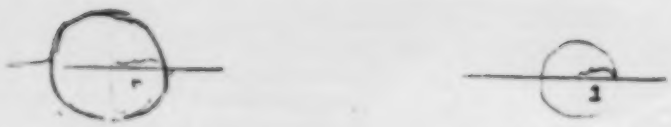
ID C1-E CLASS Reference NAME $x^2+y^2=r^2$

STMT SETTING R^2

SENT The basic form of a circle centered at (o,o) is the first to use when trying to answer general questions and solve general problems about circles

DEMON

PICTURE



MATERIALS

REMARKS

PEDAGOGY

INSPACE FORWARD ellipse, hyperbola

BACK parabola

DUAL R proof

C trig on the unit circle

APPLICATION
INTEGRATION

ID C1-E CLASS Model NAME General form of circle to standard

STMT SETTING R^2

SENT Given equation in form $ax^2+bx+ay^2+dy+e=0$, show it to be a circle.

DEMON
1. Tell about this equation $(x-2)^2+(y+1)^2=16$
2. Multiply out and combine terms. Is it still a circle?
3. Can you put it back in the original form?
4. $x^2+8x+y^2-6y+24=0$. Is this a circle? How can you show it?
5. $(x^2+8x\quad)+(y^2-6y\quad)=-24$ What is needed in parentheses?
6. Make up examples as general as possible
7. Do all equations work in R^2

PICTURE

MATERIALS Could be done for general case as a proof, but it's more than most would be interested in.

REMARKS 2. Make sure they understand x^2 and y^2 must have same coefficient

PEDAGOGY Standard

INSPACE FORWARD Quadratic systems, ellipses, hyperbolas, translations

BACK Completion of square to prove quad form and parabola

DUAL R proof of general case

APPLICATION
INTEGRATION

IDE1-E CLASS Motivational NAME Ellipse with strings attached

STMT SETTING R^2

SENT Using materials below, draw an ellipse and discover its terminology and properties.

DEMON METHOD 1. Make loop of string 8" around.
2. Draw an axis on graph paper and tack to corrugated cardboard. Add two more tacks 6 units left and right of origin.
3. Put loop around tacks and use pencil to pull taut.
4. Move pencil around paper, keeping string taut.

PEDAG Q's 1. What is figure and what do tacks represent?
2. Name some points on curve
3. What are lengths of major and minor axis?
4. What remains constant in this exercise?
5. What is length of string?
6. What is sum of focal radii?
7. Could you find length of string if it hadn't been measured?

PICTURE

MATERIALS string, scissors, corrugated cardboard, graph paper, tacks.

REMARKS 1. When trying to transfer knowledge for formal definition, remember string length is <u>not</u> same as sum of focal radii

PEDAGOGY Jacobs, H <u>A Human Endeavor</u> Ch. 6

INSPACE FORWARD Derive formula

BACK Draw a circle by this method (Move tacks closer and closer...)

DUAL

APPLICATION
INTEGRATION

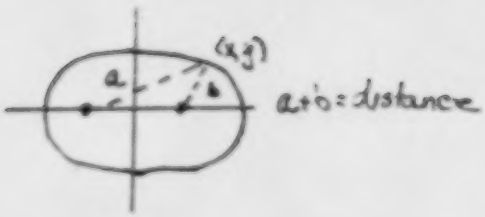
ID E1-E CLASS Start up NAME Derive equation of ellipse from definition

STMT SETTING R^2

SENT Using definition-all points, the sum of whose distance from two fixed points remains constant-derive equation for ellipse.

DEMON
1. Pick two foci(On an axis, centered on origin makes it easier)
2. Pick a distance. Is it a possible distance (>dist between foci)?
3. Pick a point on ellipse. What does definition say?
4. Form equation (Show standard form)

PICTURE



MATERIALS

REMARKS
1. Suggested foci (4,0),(-4,0), distance=10
2. Use example from ellipse with strings attached

PEDAGOGY Standard

INSPACE FORWARD hyperbola

BACK parabola, circle

DUAL C procedure for use

R Could go into formal proof but probably excessive for most classes

APPLICATION Elliptical pool table
INTEGRATION Science-orbit of planets

ID E1-E CLASS Reference NAME $x^2/a^2+y^2/b^2=1$; $x^2/b^2+y^2/a^2=1$

STMT SETTING R^2

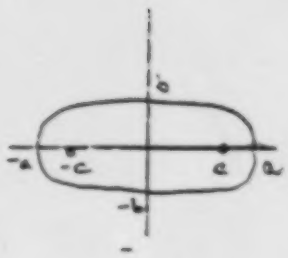
SENT The basic form of an ellipse centered at the origin is first to use to answer general questions and solve general problems about an ellipse.

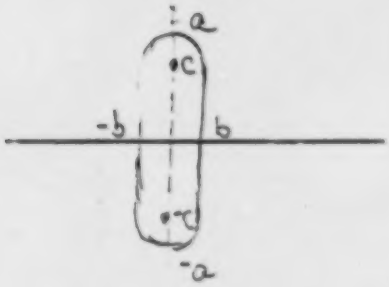
DEMON

$b^2=a^2-c^2$
foci (c,0), (-c,0)
vertices (a,0), (-a,0)
sum of focal radii 2a
y intercepts (0,b), (0,-b)

$b^2=a^2-c^2$
foci (o,c),(0,-c)
vertices (0,a),(0,-a)
sum of focal radii 2a
x intercepts (b,0),(-b,0)

PICTURE





MATERIALS

REMARKS

PEDAGOGY

INSPACE FORWARD hyperbolas, ..

BACK numerical examples, parabola, circle

DUAL C solids of revolution, rotation

R proof of formula derivation

APPLICATION
INTEGRATION

ID El-C CLASS Procedure for use NAME Equation of ellipses

STMT SETTING R^2

SENT A selection of how-to procedures for graphing ellipses from equation or information.

DEMON A. $x^2/16+y^2/9=1$ 1. How do you know it's an ellipse?
2. What are 4 axis intercepts?
3. Which is major axis?
4. What are foce?

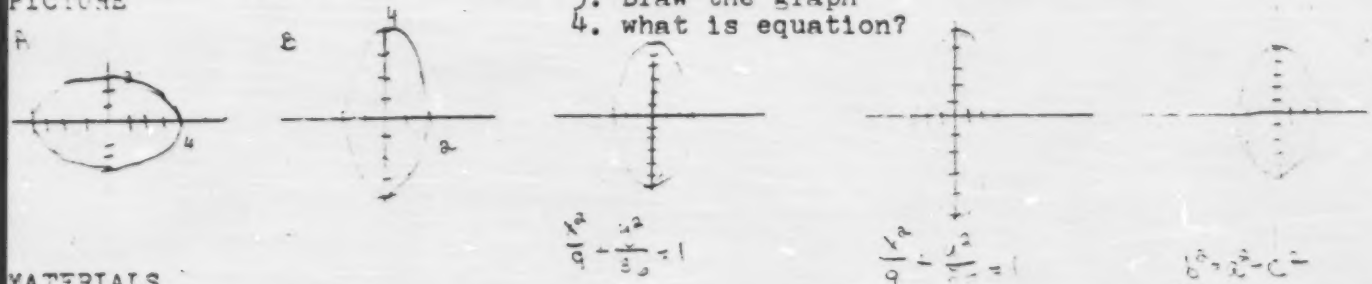
B. $x^2/4+y^2/16=1$ 1. Same questions.
2. How do A and B differ?

C. $36x^2+9y^2=324$ 1. Is this an ellipse
2. How can it be put in a form to make critical information available?

D. Major axis 10; x-int 3,-3 1. Do you know which way major axis runs?
2. What are y-intercepts?
3. Draw the graph
4. What is the equation?

E. Foci (0,3),(0,-3); sum of focal radii=10 1. Can sum help find vertices?
2. Now how find the minor axis?
3. Draw the graph
4. what is equation?

PICTURE



MATERIALS

REMARKS In C, don't work to completion

PEDAGOGY

INSPACE FORWARD

BACK

DUAL

APPLICATION
INTEGRATION

ID E1-R CLASS Transitional NAME Pythagoean relationship in ellipse and relationship of circle and ellipse

STMT SETTING R^2

SENT Show that the relationship between focal radii, foci and minor axis is Pythagorean

DEMON
1. Draw a typical ellipse and write its equation
2. draw several focal radii.
3. draw the focal radii through foci and ask for sum of lengths-2a
4. Draw focal radii equal. What is length of each
5. What is length of minor axis?
6. What is the distance from origin to focus?
7. $c^2+b^2=a^2$
8. What if c=0? a) Focus at origin
   b) $a^2=b^2$
   c) therefore it is a circle

PICTURE

MATERIALS

REMARKS Challenger: Graph $(x-3)^2/4+(y+2)^2/16=1$

PEDAGOGY

INSPACE FORWARD

BACK

DUAL

APPLICATION
INTEGRATION

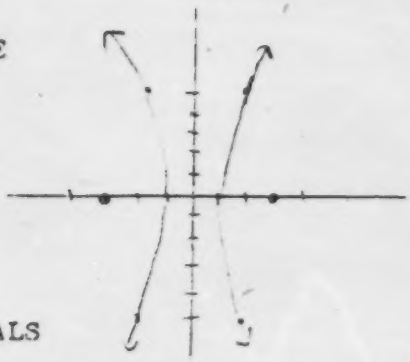
ID Hy-E CLASS Start up NAME Derivation of equation for hyperbola from definition

STMT SETTING R^2

SENT Using definition-all points the difference of whose distance from two fixed points remains constant-derive equation for hyperbola

DEMON 1. pick two foci-centering on origin makes it easier.
2. Pick the difference between focal radii.
3. Pick a sample point (x,y) on the hyperbola. What does the definition say?
4. Work equation to standard form.
5. Plot some points.

PICTURE



MATERIALS

REMARKS 1. An example that works nicely is: foci (3,0),(-3,0); difference between focal radii=2
2. Note lack of y-intercept. Discuss how to find width of hyperbola.

PEDAGOGY

INSPACE FORWARD reference form,

BACK parabolas, circles, ellipses

DUAL C procedures for use, asymptotes, inverse variation

R proof (probably not a good idea for most classes)

APPLICATION inverse variation, music-length of pipes
INTEGRATION

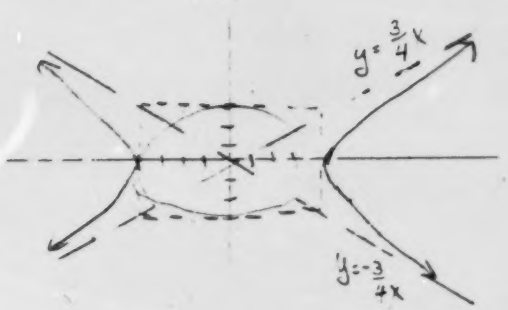
ID Hy-C CLASS Definition, procedura NAME Asymptotes

STMT SETTING R^2

SENT Students will learn definition of asymptote, how to derive and use it in a hyperbola.

DEMON
1. Using an equation for a hyperbola, plot several points to determine the general shape of the figure.
2. Graph ellipse of same equation (substitute + for -). How do the two figures compare? note that ellipse falls entirely within box formed by a,-a,b, and -b, and the hyperbola, entirely without.
3. If lines were drawn between the hyperbola and ellipse, the equation would be y=b/a x and y=-b/a x (or y=a/b x and y=-a/b x). Show inductively that as larger and larger x's are chosen, the y value approaches b/a x.
4. Pick a new hyperbola equation. Have students find and graph asymptotes, then graph hyperbola.

PICTURE



Ellipse $\frac{x^2}{16}+\frac{y^2}{9}=1$

Hyperbola $\frac{x^2}{16}-\frac{y^2}{9}=1$

MATERIALS

REMARKS 1. Use an intuitive notion of limit (e.g. halving distances, close enough for all practical purposes)

PEDAGOGY

INSPACE FORWARD formal concept of limit

BACK tangent angle as limit of secant angle

DUAL

APPLICATION
INTEGRATION

ID Hy-E CLASS Reference NAME $x^2/a^2-y^2/b^2=1$; $y^2/a^2-x^2/b^2=1$

STMT SETTING R^2

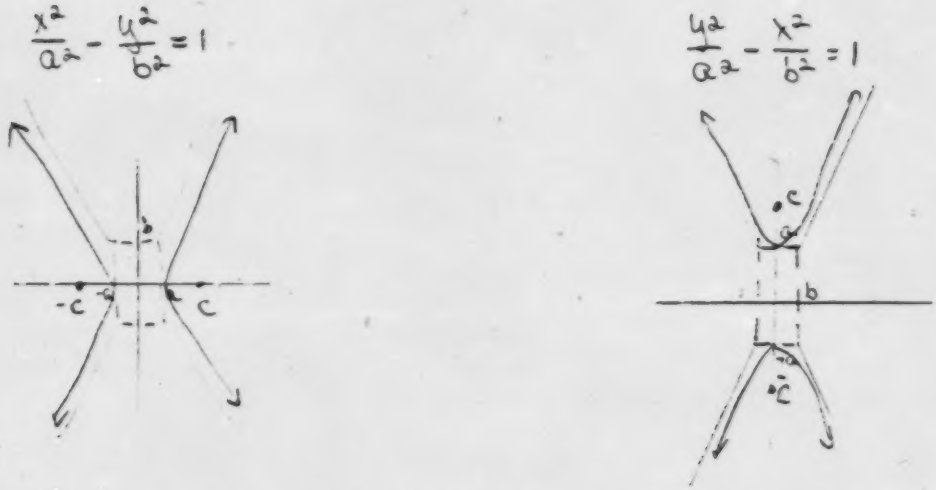
SENT The basic form of a hyperbola centered at the origin is the first to use when trying to answer general questions and solve general problems about hyperbolas.

DEMON

$b^2=c^2-a^2$
foci $(-c,0),(c,0)$
verticies $(-a,0),(a,0)$
asymptotes $y=b/a\ x; y=-b/a\ x$

$b^2=c^2-a^2$
foci $(0,-c),(0,c)$
vertices $(0,-a),(0,a)$
asymptotes $y=a/b\ x; y=-a/b\ x$

PICTURE



MATERIALS

REMARKS 1. Point out that "a" need not be the larger number

PEDAGOGY

INSPACE FORWARD number examples

BACK parabola, ellipse, circle

DUAL C procedure for use

R proof

APPLICATION
INTEGRATION

ID Hy-C CLASS Procedure for use NAME Equations of hyperbolas

STMT SETTING R^2

SENT A selection of how-to procedures for graphing ellipses from equation or information.

DEMON

A. $x^2/4-y^2/9=1$
1. How do you know it's a hyperbola?
2. What are the vertices? How do you know which axis to use?
3. Where are the asymptotes?
4. Where are the foci?

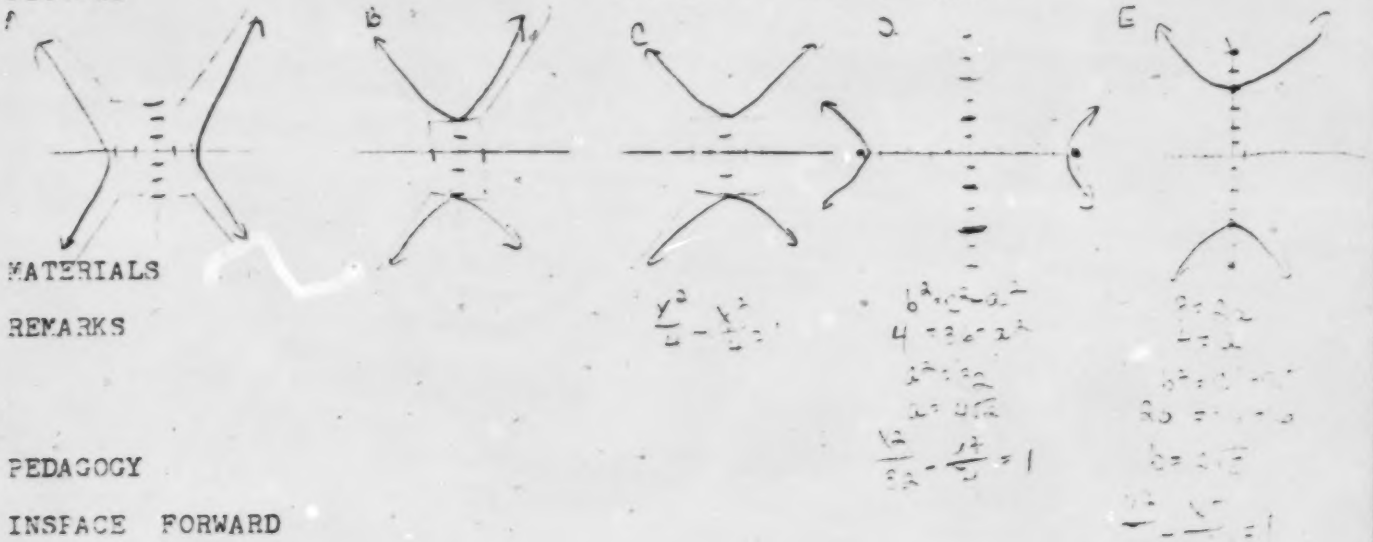
B. $y^2/4-x^2=1$
1. Same questions.
2. How do A and B differ?

C. $4y^2-4x^2=16$
1. How do you get the equation into an appropriate form for graphing and finding important points?

D. Foci at (6,0),(-6,0) lgth of conjugate axis=4
1. How can you find vertices?
2. What are asymptotes?
3. Draw the graph.
4. What is the equation?

E. Foci (0,6),(0,-6) /diff of focal radii/=8
1. Can difference help you find the verticies?
2. How find the conjugate axis?
3. Draw the graph.
4. What is the equation?

PICTURE



MATERIALS

REMARKS

PEDAGOGY

INSPACE FORWARD

BACK

DUAL

APPLICATION
INTEGRATION

APPENDIX 2

THE STRUCTURE OF KNOWLEDGE IN COMPLEX DOMAINS

Edwina L. Rissland
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Table of Contents

I. INTRODUCTION

Objectives
Structure and Rep

II. A CONCEPTUAL FRAMEWORK

Structure: Items, Relations and Spaces
Mathematical Knowledge
Structure: The Notion of Dual
Taxonomies
Context
Representation Frames

III. AN EXAMPLE

IV. IMPLICATIONS FOR TEACHING and LEARNING

This work supported in part by NIE grant G-80-0096. Any opinions, findings, conclusions or recommendations expressed in this report are those of the authors and do not necesarrily relfect the views of the U.S. government.

I. INTRODUCTION

Objectives

Learning, teaching and understanding are complex tasks in complex domains like mathematics. Any teacher or student knows that there is rich complexity to the subject of conic sections or continuity. Knowing such domains is much more than knowing formal statements of definitions, theorems and proofs: there are examples; methods; exercises; pictures, graphs and diagrams; rules of thumb; metaphors and analogies; folksy or informal formulations of ideas; a sense of what is important; a sense of what should follow what. This complexity exists whether the purpose is to learn or to teach.

One objective here is to present a conceptual framework in which to describe and understand knowledge in complex domains. Another is to show how such a framework can aid teaching and learning in them.

There are several underlying themes. First, understanding a domain requires a great familiarity with its connections. Polya and Szego [1972] said eloquently:

> There is a similarity between knowing one's way about town and mastering a field of knowledge: From any given point one should be able to reach any other point. One is even better informed if one can immediately take the most convenient and quickest path from the one point to the

> other. If one is very well informed indeed, one can even execute special feats, for example, to carry out a journey by systematically avoiding certain paths which are customary...
>
> There is an analogy between the task of constructing a well-integrated body of knowledge from acquaintance with isolated truths and the building of a wall out of unhewn stones. One must turn each new insight and each new stone over and over, view it from all sides, attempt to join it on to the edifice at all possible points, until the new finds its suitable place in the already established, in such a way that the areas of contact will be as large as possible and the gaps as small as possible, until the whole forms one firm structure.

Second, being aware of what one knows and how one comes to know it are important ingredients of expertise. Disambiguation and explication of the process of knowing enable one to be a smarter student or teacher, as Papert and others have often pointed out [Papert 1971, 1980][Buchanan and Headrick 1970].

Thus, we are engaged in <u>epistemology</u>, the study of knowledge. The epistemology of complex domains includes the study of their content, structure, representation, and use. There are many dimensions and levels to epistemology. There is study of the domain itself, for example, through analysis of its content and structure. There is the epistemology of one domain with respect to different conceptual frameworks; and the epistemology of one compared with another with respect to the same framework. There is the epistemology of epistemologies, that is, the study and comparison of frameworks.

In this paper, we will discuss not only the epistemology of particular domains, but, more importantly, also a general

epistemological framework. Our approach reflects concerns and methodologies from cognitive science and artificial intelligence (AI). Important contributions of AI are the tools of information processing and knowledge representation [Feigenbaum and Barr 1981] [Winston 1977]. To be able to use them means that not only should we know the knowledge, but also what we want to do with it.

Our analysis is based upon the experience of the author and others as teachers and learners. The structural notions presented here seem natural to students. Several students, e.g., in calculus, have found that using such notions helps to give them understanding that is different from that of acquiring a large bag of tricks. Our conceptual framework is currently being applied to investigations in learning and understanding probability [Myers 1980], discrete mathematics for computer science [Wegner 1981], and high school mathematics [Davis 1977].

One of the themes running through our work is the importance and interaction of examples with other knowledge such as concepts and results. The role played by examples in the evolution of mathematical concepts is beautifully described by Lakatos [1976]. Examples play a critical role in learning and concept formation by machines [Winston 1975] [Lenat 1976] as well as people [Polya 1965][Collines 1979]. Examples can also be a very effective means of presenting new technical ideas [Wegner 1980].

Of course, attention to and understanding of the topic of

heuristics and its role in discovery are due to the Polya. His well-known books are full of insight, e.g., [Polya 1965]; they also provide many examples of the kind of knowledge we wish to explicate further.

Structure and Representation

In this paper, we distinguish between "structure" and "representation". Structure is something inherent in the knowledge domain which can be teased out by examination of the domain and how it is used. Representation is the way in which we encode the knowledge and structure.

We can see structure, or different aspects of it, when we examine the domain. The structure discerned is clearly influenced by our purposes, expectations and point of view [Kuhn 1970], [Bruner 1973]. For instance, if we work with situation-action rules, such rules will jump out as we examine a domain (e.g., [Greeno 1978]). With different purposes, we emphasize different aspects. For instance, if our purpose is to study logical development, we attend to deductive aspects like definitions, theorems and proofs; if our purpose is to study expository style, we attend to rhetorical structures. In this paper, our purpose is to understand the knowledge better, so as to improve our skills as teachers and learners; and to enhance our understanding of understanding.

Often there are many ways to represent the same knowledge; the one chosen also depends on our perferences and purposes -- on what we are going to use such knowledge for. For instance, researchers in logic and theorem proving often use the predicate calculus. We choose the representation scheme that best suits our purposes;

"best" in the sense of conceptual clarity, ease of encoding or use. For instance, representation schemes like "frames" [Minsky 1975], "scripts" [Schank and Colby 1973], "schemas" [Rumelhart and Ortony 1977] and "semantic nets" [Quillian 1968], [Brachman 1979] make it easy to encode clustering, typing, and connectivity.

An underlying viewpoint in our approach is that chunking of knowledge is very important. In fact, what might distinguish an expert from a novice is the kind and degree of chunking used in thinking about a domain. As one becomes more expert one is able to use bigger chunks, more encompassing and general. For instance, the expert mathematician sees a proof as a coherent whole, perhaps with substructure [Rissland 1978a], while the novice may see only the succession of individual proof steps. The novice's view is very <u>local</u>; the expert's can be <u>global</u> as well. Experts can switch the level of description easily and naturally.

This pattern is also found developmentally: young children often cannot summarize a story; they retell it line-by-line; older children can present a synopsis. Adults often use more global organizations like "scripts" [Schank and Abelson 1977].

II. A CONCEPTUAL FRAMEWORK

Structure: Items, Relations and Spaces

There are several kinds of structure in the knowledge of complex domains like mathematics:

(1) strongly bound clusters of information, like the statement of a theorem, its name, its proof, a diagram, an evaluation of its importance, remarks on its limitations and generality. In this paper, we call such clusters *items*.

(2) *relations* between items, like the logical connections between results, such as *predecessor* results on which they depend logically and *successor* results which depend on them.

(3) sets of similar types of items related in similar ways, like proved results and their logical dependencies. Such a set of items and relations constitute a *space*, for instance, *Results-space*.

What distinguishes a "space" from a "set" is the prominence of the relations. The structure of a complex domain like mathematics contains not just one but many spaces, each of which describes a different aspect of knowledge. In mathematics, for instance, as found in lectures and textbooks, e.g., [Rudin 1964], there are

spaces of:

results: lemmas, propositions, theorems, corollaries, etc., related by logical dependency, i.e., how one result is deduced from others;

examples: illustrative situations and cases related by constructional derivation, i.e., how one is built from others;

concepts: formal ideas like definitions, and informal ones like heuristic principles related by conceptual dependency, i.e., how one concept is defined or presented in terms of others;

exercises: exercises and unresolved conjectures related by how one exercise is generated from others, e.g., as a subproblem that would contribute to solution of a larger problem;

procedures: procedures and methods related by procedural dependency, i.e., how one procedure is composed of or depends on others, as in program subroutines;

strategies: control methods, doctrines or heuristics related by how one strategy engenders or contributes to the practice of others;

goals: goals and purposes related through dependencies of super- and sub-goals [Newell and Simon 1972], i.e., how one goal

engenders or contributes to the attainment of others.

In our freedom to use many spaces, we have departed somewhat from the usual semantic net approach in AI since we use many nets, i.e., spaces, simultaneously. Each space represents a different "cut" through the domain. In summary, if the same type of relation can be seen in a set of items, then we can describe it as a space.

Mathematical Knowledge

While all of the above mentioned spaces, and probably others, would be needed to construct an exhaustive epistemological framework for describing complex domains in general, a significant part of the knowledge in domains like mathematics and computer science can be captured in the following spaces: Results, Examples, Concepts, Exercises, and Procedures.

Results

The set of proved results and their logical dependencies constitute Results-space. For instance, in plane geometry [Jacobs 1974], the "Isosceles Triangle Theorem", which states that if two sides of a triangle are equal, then so are the angles opposite them, depends logically upon the side-angle-side postulate which is used in its proof; a corollary result depending on this theorem states that an equilateral triangle is equiangular.

Examples

An important aspect of examples is that one builds examples from other examples. The relation of constructional derivation emphasizes this and allows the collection of examples to be coherently organized. For instance, in order to show that not all second degree equations represent conic sections, one can modify the example of the equation of a circle

x**2 + y**2 = 4

to x**2 + y**2 = -4. Thus, the second example item would point back to the first, and the first would point forward to the second.

Another instance of the construction of examples is the derivation of the example y = xsinx near the origin

<insert picture>

from the reference examples y = -x, y = x and y = sinx.

<insert pictures>

by squeezing the oscillations to occur between the lines y=x and y= -x.

The generation of examples is an interesting study in itself [Gelbaum and Olmstead 1964], [Steen and Seebach 1970], [Rissland 1980].

Concepts

Concepts include formal and informal ideas. A formal idea is a precisely stated notion like a definition, e.g., the familiar

"epsilon-delta" definition of continuity given in calculus. An example of an informal idea is the paraphrase of continuity as "not lifting the pencil from the paper". It is not a bona fide mathematical definition but it is an idea that is frequently used. Informal ideas also include individual heuristics of two types: mega-principles and counter-principles. A mega-principle is a "big" idea that either is an informal truth, such as "Continuous functions have no gaps or breaks", or suggests a good line of attack, such as "Try factoring the polynomial into linear factors". A counter-principle, on the other hand, summarizes troublesome points or offers a warning, like "Be careful about dividing by quantities that might be zero" or "Don't forget to calculate the new dx".

Concepts can be related by dependencies of definition or pedagogy. A definition can be definitionally dependent on another if it uses the other in its formulation; for instance, the limit definition of continuity uses the concept of "limit". Pedagogical dependency is more subjective: we feel that one concept should be learned or introduced before another. Some authors map out the conceptual dependencies of topics and chapters in their books e.g., [Dunford and Schwartz 1958], [Royden 1963].

Exercises

Exercises play an important role in helping a student. They can be related in several ways, as in the use of the solution or

method of one exercise to pose or solve another. For instance, solving the cubic equation

$$x^3 - 1 = 0$$

involves solving the quadratic equation

$$x^2 + x + 1 = 0$$

More complicated, and often subtle, relations between exercises are found in higher level mathematics courses like real analysis.

Procedures

In many domains, like computer science and applied mathematics, there are large collections of methods and procedures. Procedures can be related by the dependency of one procedure on another. This relation is important in analyses of skills like arithmetic, where for instance addition is a predecessor skill for multiplication. The ordering is often strongly held by subjects [Davis and McKnight 1979] and is often related to student "bugs" [Seely-Brown and Burton 1977].

The dependencies often organize procedures into hierarchy. For instance, the procedure to find the roots of a cubic equation like the one above is composed of the procedures to factor it as

```
(x - 1)(x**2 + x + 1)
```

and then to pick off one root, 1, by observation, and, finally to compute the other two by using the quadratic formula. For cubic equations with not so nice a form, the solution procedure might include trying to guess a root, which itself involves procedures, and then proceeding as before.

Representation-graphs

The dependency relations in the spaces allow us to treat the spaces as directed graphs where the nodes represent items and the arrows, the relations between them; the direction shows the inherent predecessor-successor ordering. In this way, we have have several graphs: results-graph, concepts-graph, examples-graph, etc.

Thinking of the spaces as directed graphs is a useful conceptual aid since it allows us to picture them easily and from the picture to discern connections and the overall complexity of the domain. In fact, the use of the word "space" is itself an aid since it calls to mind a mathematician's idea of space with its notions of coherence and distance. The graphs are a way to picture the representation spaces that represent the structure of the domain; the graphs themselves are not a part of the domain.

Structure: The Notion of Dual

An item is related to other items in its representation space through the space's dependency relations. In addition, an item is related to items in the other spaces. The notion of dual concerns these inter-space relations.

As an illustration, consider results, examples, and concepts, which together constitute a large part of the knowledge in many mathematical domains. Results are connected to examples and concepts as well as to other results. The dual items of a result include: (a) the examples motivating and illustrating it; (b) the concepts needed to formulate it and those derived from it.

Similarly examples and concepts are connected to items outside of their spaces. The dual items of an example include: (a) the results illustrated by it and proving things about it; (b) the concepts illustrated and suggested by it. The dual items of a concept include: (a) results leading to its formulation and those proving things about it; (b) examples motivating and illustrating it.

Examples, results and concepts also have dual items from the spaces of procedures, goals, exercises, etc. Exercises are closely tied to the procedures, examples, concepts and results used in their solution and those that the exercise illustrates and affords practice with. In many high school texts, the relations

between exercises and worked out examples are particularly close.

Strategies, goals and procedures are often strongly connected because the strategies are dual to the goals that they are useful for achieving; the procedures are ways to implement or carry out the strategies. In programming, this is often called "pragmatics" as compared with the syntax and semantics.

Thus each item is related to certain items from the other spaces: we call the set of those items, its *dual*. Duals themselves have structure: some dual items tend to precede the item in teaching and learning, others, succeed it. For instance, motivating examples for a result come before the result; other examples, like those showing the limitations of the result (e.g., that the converse is not true) come after it. Thus, we can distinguish pre and post duals. These of course may depend on pedagogical and cognitive style.

The power of the dual notion is that it allows us to associate items that might not be closely related with regards to in-space relations. For instance, two concepts might be widely separate in Concepts-space (e.g., because they are taught many chapters apart), but are closely related in the dual sense in that they share many examples. An important part of understanding is building up such associations; assigning exercises which draw on both concepts is a way to help students establish such dual relations.

Taxonomies

Another important component of knowledge is the knowledge that not all items serve the same function in learning and understanding. For instance, experts (teachers and learners) know that certain perspicuous ("start-up") examples provide easy access to a new topic, that some ("reference") examples are quite standard and are always exhibited as illustrations, and that some examples are anomalous and don't seem to fit into one's understanding.

We can develop a taxonomy of items based upon how we use them to learn, understand and teach a domain. In discussing Concepts-space, we have already mentioned different kinds of concepts. Briefly, some important taxonomic classes of items are:

(1) Results

(a) basic results: basic, first proved results;
(b) key results: major, frequently used results;
(c) culminating results: goal results;
(d) transitional results: intermediate logical stepping-stones;
(e) technical results: results establishing technical details;

(2) Examples

(a) start-up examples: perspicuous, easily understood and presented cases;

(b) reference examples: standard, ubiquitous cases;

(c) counter examples: limiting, falsifying cases;

(d) model examples: general, paradigmatic cases;

(e) anomalous examples: exceptions and pathological cases;

(3) Concepts

(a) definitions: formal definitions and specifications;

(b) informal paraphrases: informal formulations;

(c) mega-principles: "big" ideas like positively oriented heuristics;

(d) counter-principles: heuristic warnings;

One can also distinguish kinds of procedures, exercises, and goals. Such a taxonomy is not an exclusive classification since an item can serve more than one role; for instance, some start-up examples become reference examples as one learns more in the domain. It is also surely not exhaustive.

Worth Ratings

Not all items are of equal importance. One can use a worth rating, such as a scheme of *'s (as in the Michelin guidebooks):

- * an interesting item, worth noticing;
- ** an important item, worth a "stop";
- *** a very important item, worth a "detour";
- **** an extremely important item, worth a "journey" in itself;

The worth rating and the taxonomy are related in that certain classes tend to be worth more than others; culminating results tend to have three and four-star ratings, whereas technical or transitional results might not even rate one star.

Just as items can be given a worth rating, so too can relations. For instance, the dual relations of examples to a given result probably have varying degrees of worth and importance: a telling counter-example, say to the converse of a theorem, is probably much more important as a dual example than an example which is just another instance of the theorem. Tagging the relations provides a way of describing the worth of an item relative to another item. Thus in addition to the "global" worth of an item with respect to the entire knowledge domain, items can have "local" relative worth. In this way, for instance, a 1-star example which is usually not very important can be recognized as very important in relation for a particular item.

Context

In describing a domain, we must also include contextual information such as assumptions about what domains precede it and which items can be taken as known or axiomatic; these might be culminating results of a predecessor domain. Also important is the state of knowledge of the learner in studying the domain, e.g., prerequisite skills and material.

In mathematics the "same" set of items can be studied in several mathematical contexts or settings: one can study operators in finite dimensional vector spaces (i.e., matrices) or in Hilbert space or in normed linear spaces. The settings themselves can be organized into a space with a generality relation or what AI researchers refer to as "is-a" or "ako" ("a kind of") hierarchies.

Since our purpose in this paper is to concentrate on a domain within a given context, we shall not dwell upon the influence of setting other than to say that it is important. We will keep our discussion localized to one context.

Representation Frames

In our representation of an individual item we include the kinds of information described above. We use the "frame" format [Minsky 1975] where slots can contain declarative, procedural or relational information.

Items have a NAME, like "continuity". The CLASS of an item is its taxonomic class. The WORTH slot contains an evaluation of worth, perhaps in terms of zero to four stars. Its SETTING is the context in which it is valid or presented.

An item can be presented in more than one way: as a declarative STATEMENT, in terms of its derivation or PROCEDURE, or by a PICTURE or diagram.

For instance, the declarative statement of an example states what the example exemplifies; its procedural aspect describes its construction; its picture is a schematic diagram or plot or kinetic sequence of pictures. The declarative aspect of a result is a statement of its hypotheses and conclusions (e.g., in "if-then" form); its procedural aspect is its proof. A concept can be stated declaratively or in terms of a procedure that implements it or that provides a way to test whether an item is an instance of it. For instance, the "derivative" can be defined in terms of limits or in terms of differentiation procedures.

The declarative aspect of a procedure is a statement that says what it does; its procedural aspect is its code. The declarative aspect of an exercise is the problem statement; the procedure could be its solution.

Relational information is encoded in slots containing pointers to other items. IN-SPACE pointers point BACK to predecessor items and FORWARD to successors. DUAL-SPACE pointers point to dual items.

The pointers, or sets of pointers, themselves have structure. We have already mentioned the pre- and post- distinction for dual pointers and that pointers can also have worth ratings. Pointers can also be partioned into clusters of dual items that can be thought of as a group bound to the item (e.g., a cluster of examples varying slightly only in their numerical values) or sets of predecessor items that are sufficient to derive the item (e.g., there may be more than one way to derive a proof from other proved results).

An item can have REMARKS such as NOTA BENE's (NB) and CAVEATS which point out particularly noteworthy or critically limiting things about the item. An item might have PRAGMATIC data telling what the item is good for. In addition an item has PEDAGOGICAL information regarding how and when to present it, like materials needed for a demonstration of it, or leading questions to ask about it; BIBLIOGRAPHIC data like references to books, articles, films; APPLICATIONS data pointing to real-world applications. OTHER data might include historical remarks.

III. AN EXAMPLE

In this section, we map out and discuss an important mathematical domain, that introducing the notion of "function" in high school algebra.

This extended example will illustrate how one can distil the ingredients of knowledge from a standard mathematics text and represent them in the conceptual framework developed in the previous sections.

It should be said that in extracting this topic for illustration we lose some of its richness; it will seem shallower and less connected than it would in the context of the entire text or course. However, even in the few pages of text examined, there is a surprising amount of material.

The examples will be taken from a widely used text Dolciani, Wooton, Beckenbach and Sharron [Dolciani 1968]. Our example deals with the topic of functions and relations (Sections 4-1 and 4-2). It assumes background about sequences of real numbers, which was covered in the previous chapter.

Section 4-1 begins with a discussion of the example of the negative, odd integers, which can be thought of as constructed from the important reference example of the odd integers:

-1, -3, -5,..., 1-2n, ...

Using this example, the authors introduce the notion of "pairing" and "ordered pair"; the third number in the sequence is paired with -5, through the formula 1-2n, which evaluates to -5 for n=3, and, of course, by simple counting.

The authors then define the concepts of "coordinates", "components" and "equality of ordered pairs". Equality of ordered pairs is then illustrated with

(3, -5) = (3, 1-2*3).

Inequality is illustrated with

(3,-5) =/= (-5,3).

Thus the first pairs serve as an illustration (post-dual example) to the "equality of ordered pair" concept; the second, a derivate of the first, as a limiting counter-example.

The discussion returns to the primary topic of functions:

> "Sequences are special examples of an important mathematical concept, called a <u>function</u>. A <u>function</u> <in red boldface> is any pairing of the members of one set (the <u>domain</u> <red boldface>) with the members of another set (the <u>range</u> <red boldface>) so that each member has <u>exactly one</u> partner in the range. Thus, the infinite arithmetic sequence
>
> -1, -3, -5,...
>
> is a function whose domain is {the positive integers} and whose range is {the negative odd integers}."

Thus, using the negative odd integers as a start-up example, the authors have introduced the definition of function. If we were to fill in representation frames for the negative odd integers example and the function concept (as presented so far), they would

be:

NAME: Odd negative integers
CLASS: start-up
WORTH: *
SETTING:

DECLARATIVE STATEMENT: This example introduces the concept of function through the concept of ordered pair.

PROCEDURAL STATEMENT: a<sub n> = 1-2n

IN-SPACE POINTERS: BACK: Odd integers
FORWARD:

DUAL-POINTERS: PRE-CONCEPTS: Sequence
POST-CONCEPTS: Ordered pair, Function

(Note that into the frame we are entering only the knowledge presented in the text. Putting in additional knowledge that reflects our own understanding is quite a different exercise.)

NAME function
CLASS definition
WORTH ****
SETTING

DEC STMNT: A <u>function</u> is any pairing of the members of one set (the <u>domain</u>) with members of another set (the <u>range</u>) so that each member has exactly one partner in the range.

IN-SPACE POINTERS: BACK: Ordered pair
FORWARD:

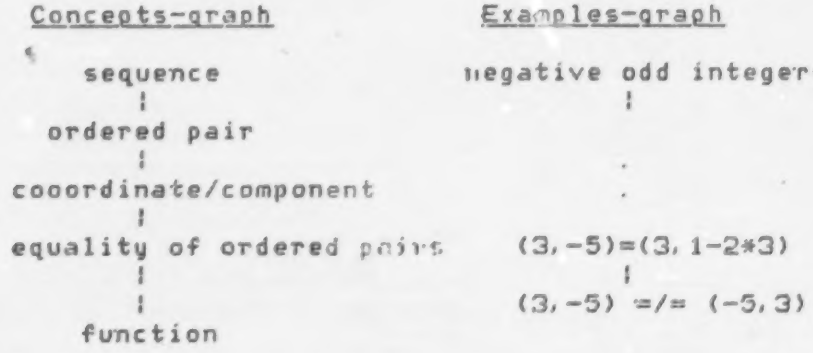
DUAL-POINTERS: PRE-EXAMPLES: Odd negative integers
POST-EXAMPLES:

NB: Each member of the domain has <u>exactly</u> <u>one</u> partner in the range.

(We have set the worth at four stars because Dolciani et al have used the word "important".)

To build a complete representation, we would also need to create frames for the concepts of "ordered pair", "coordinate/component", and "equality of ordered pairs". The frame for "ordered pairs" would have a PRE-EXAMPLES DUAL pointer to the "negative odd integers" example which was used to motivate it; it would have FORWARD IN-SPACE pointer to the concept "coordinate/component" which in turn would have a FORWARD pointer to the concept "equality of ordered pairs", which would point BACK to "coordinate/component" which in turn would point back to "ordered pair".

Interconnections among the examples and concepts introduced thus far is shown in the following fragments of the examples- and concepts-graphs:



Next, the terminology of value and belong to is explicated. We have the following statement which the authors italicize for emphasis:

> "Thus, you can think of a function as a set of ordered pairs in which different ordered pairs have different first coordinates."

This paraphrase can be thought of as the first variation on the theme of function; we could represent it as a concept which follows from the function concept and inherits its slot values (unless otherwise noted) by default from its predecessor, the "function" concept. Alternatively, in our representation frame, we could introduce a slot for PARAPHRASE.

Next, we begin to acquire some procedural knowledge about functions:

> "One way to specify a function is to show its ordered pairs in a table with the members of the domain named in one column (or row) and the corresponding values of the function in another column (or row)."

A simple example illustrates the function table idea:

Domain Element	Function Value
-1	-1
-2	-1
3	1
4	1

Note that the authors are combining the first few integers, particularly +1 and -1, with a sprinkling of minus signs, to generate an example. One could say that this is an instance of the "meta"-heuristic of "using plus and minus one's" for generating examples.

Next comes a nota bene remark on the preceding example and the function concept:

> "Notice that this function does not have a different value corresponding to each different member of its domain."

This NB deserves comment and encoding as a counter-principle, which we could name CP(not 1-1). It is closely connected with the "exactly one" NB of the "function" concept, which also is acquiring the status of a counter-principle, which we name CP(exactly one). As the authors state in their teacher's manual:

> "This is a good place to make sure that students understand that although each element of the domain of a function is paired with exactly one element in the range, there is no restriction as regards ... an element in the range."

The preceeding function table example is then re-presented as an example of specifying a function by a set of ordered pairs:

$$\{(-1, 1)\ (-2, -1),\ (3, 1),\ (4, 1)\}$$

The section's start-up example (negative odd integers) is further used to illustrate the table procedure. It is also used to introduce specification of a function in terms of a formula and the use of a function symbol:

> "The formula a<sub n> = 1-2n, n <e> {the positive integers} provides a compact way to show pairings in the sequence and suggests a convenient way to indicate function pairings in general. However, let us replace the symbol "a<sub n>" with the function symbol
>
> a(n), read 'a of n'..."

At this point there is another warning to the teacher:

> "Be sure that students do not call the formula a function. The formula does, however, specify or define a function over (in this case) (the positive integers)."

This is really another counter-principle, which we name CP(formula vs function).

To update our function frame, we would now add the function table and formula information to the procedural statement, PROC STMNT slot, another example pointer, and pointers to the CP(not 1-1), CP(exactly one), and CP(function vs formula) counter-principles.

We now come to "Example 1", the first explicitly labelled example, the standard reference example of the absolute value function over the domain of the real numbers. A discussion of the implicit definition of the domain, i.e., the domain of definition, is presented briefly.

From these remarks, we also learn that the tacit setting for all of the discussion is the real numbers, $\mathcal{R}$.

"Also, it is agreed that the domain, unless otherwise specified, consists of those real values of x for which the formula provides a unique real value of f."

This "agreement" is illustrated with the function

$$f(x) = 1/x$$

Note that the authors have begun to use the letter "f". This seems unremarkable to anyone who has studied functions; it is part of the tacitly held knowledge that "f's", "g's" and "h's" are often used for functions, and "a's" for sequences. Furthermore, the second function (1/x**2), while not explicitly derived from the first (|x|), is not unrelated to it, because the absolute value function can be defined in terms of the square root:

<sq rt>x**2

Thus, the authors are really using examples coming from a cluster of examples around the x**2 example. This knowledge is not explicitly present in the section of text under examination, but is present in the larger fabric of algebra and mathematics in general. The connections probably help to make the discussion more coherent, at least to the authors and the teachers.

Example 2 is, in fact, the important reference example x**2, which is ubiquitous in discussions about functions, polynomials, parabolas, etc. The example is presented as:

$$h = \{(x, h(x)) : h(x) = x^{**}2\}$$

It is used as an illustration of the concepts of function, domain, range, and domain of definition; thus its frame would have pre-dual pointers to these concepts and their frames would have post-dual pointers to it.

The next example, Example 3, is a counter-example serving to differentiate between the concept of "set of ordered pair" and "function". It can also be thought of as a counter-example to highlight the "exactly one" counter-principle or nota bene remark (NB) of the function concept:

> "Let g be the of all ordered pairs of real numbers such that g = {(x,y) : xy**2 = 1}."

(Notice the use of second power again.)

This counter-example for "function" serves as a start-up example for "relation", which is then defined in terms of the "order pair" concept. This concludes Section 4-1.

Thus at the conclusion of the text of this first section on functions, we have several elements in Examples- and Concepts-spaces, and none in the others (although we have begun to build up knowledge of procedures, like graphing, which could be represented as procedure-items when they acquire "critical mass".) We have spent most of the discussion on the "function" concept and a large part on the "negative odd integers" example.

Thus, at in a very few pages (three, to be exact), we have established the "function" concept, whose frame at this point would look like:

NAME function
CLASS definition
WORTH ****
SETTING R

DEC STMNT: A function is any pairing of the members of one set (the domain) with members of another set (the range) so that each member has exactly one partner in the range.

IN-SPACE POINTERS: BACK: Ordered pair *
FORWARD:

DUAL-POINTERS: PRE-EXAMPLES: Odd negative integers
POST-EXAMPLES:

NB: Each member of the domain has exactly one partner in the range.

In a high school text, exercises play an important role. The exercises for Section 4-1 are used to provide the student practice with the new concepts. Thus they serve mostly a dual role. A few harder exercises (in group "C") serve as nice introductions to other topics by encouraging the student to discover new knowledge.

The exercises are organized according to the concepts which they treat: i.e., by dual relations to concepts. In other sections, the exercises are often dual to examples in the sense that they ask the student to work through an example or procedure.

In the second section on functions (4-2), the authors begin by discussing the notion of "mapping", introduced with an example of mapping in the everyday sense.

This notion is then pulled into the function discussion:

> "You can think of any function as a mapping of its domain into its range."

Thus the authors have introduced the second important variation of the function concept, which we could call the "function as mapping" mega-principle.

The important schematic of the "mapping diagram" quickly follows:

Domain	Range
5	1
10	2
20	2
30	3

Domain		Range
5	---->	1
10	---->	2
20	---->	
30	---->	3

<draw mapping diagram>

Thus the procedures of making a function table and a mapping diagram are linked. Next comes the graphing procedure:

> "Another way of picturing a function depends on the fact that ordered pairs of real numbers can be graphed in a plane."

There is next a digression about graphing and coordinate systems. Before giving examples, the authors state the graphing procedure for a function and the definition of "graph":

> "To picture a function in the coordinate plane, plot the graphs of all the ordered pairs of the function. The set

of points obtained is called the graph of the function."

Note that the "graph" concept has been defined procedurally as the result of the "graphing procedure".

Drawing a graph is illustrated with Example 1 of Section 4-2

$$1 - |x|,$$

which is closely related to Example 1 of Section 4-1, the absolute value function

$$f(x) = |x|.$$

Before Example 2,

$$g: x \text{ ---> } 3x-5, \quad x \in \{1,2,3,4\}$$

the authors introduce another way of representing a function as a "mapping statement" that involves the use of the arrow. Example 2 is not particularly noteworthy, except that once again it is built around the integers 1, 2, 3, 4.

Example 3, $L = \{(x,y) \mid x=4\}$, is used as a counter-example to distinguish between the "function" and "graph" concepts. It is dual to "exactly one" counter-principle. It is also an example for the "relation" concept. Its graph, which lies on a vertical line, gives a pictorial representation for the "exactly one" CP.

The oral exercises for this section are quite nice, because they exercise both the mapping diagram and graph representations for functions. Included are examples that are not functions. There are several "favorite" reference examples for the concepts of

one-to-one, onto, bijection, continuity, differentiability, etc. For instance, the graphical exercises (nos. 5-16) contain: (1) a function composed of other functions (#9); (2) a function with a "jump" discontinuity (#10); (3) a graph which is a well-known figure, but not a function, i.e., a circle (#11); (4) a step function (#13); (5) a non-function of x, but a function of "y" (#15); (6) an absolute value function, -|x| (#12); and the absolutely standard parabola y=x**2 (n 14).

Thus in two short sections there has been a tremendous amount of knowledge clustered about the function concept.

While it is not reasonable to expect anyone to analyze or map out all of a mathematics text in this detailed a fashion, it is useful to examine a section or two in this way to highlight the amount of knowledge that is presented or tacitly assumed, and the degree of interconnectedness of it. Such an exercise also serves to highlight the complexity of the task of learning and understanding new knowledge.

IV. IMPLICATIONS FOR TEACHING AND LEARNING

In addition to representing the knowledge contained in textbooks, the conceptual framework can be used to organize what we know about a subject. The process of filling in the frames prompts us to recall and disambiguate knowledge. It can also help structure and represent lessons, which for instance, could be thought of as a sequence of frames.

The framework can be used explicitly to help students understand or explore a domain, for instance, by asking them to map it out or fill in frames. The frame implicitly contains questions to be asked about an item (e.g., "How important is it, and why?"). Students, at least at the college level, enjoy the mapping process and find it helpful to their acquisition of understanding (e.g., when reviewing the subject). In some ways the mapping process is more important than the representation finally produced. Exercising their knowledge by directing their attention to important items and relations gives focus to their efforts at synthesis. Using specific questions that probe and prompt understanding, such as found in [Rissland 1978], gives the students a way to <u>actively</u> pursue understanding, a process often expected to happen magically.

Thus the framework suggests new homework and test questions, like those to probe dual connections (e.g., "Give two examples of this concept: a standard case and a not so "nice" case."). Students

are not asked to give enough examples, much to their detriment since generating examples forces them to understand the involved concepts, results, etc. One can encourage them to explore duals in another way by asking why two items, e.g., theorems, are related (even thought they are found in different chapters). Such questions will encourage students to become active investigators.

For teachers, the conceptual framework offers guidance in choosing items for presentation. For instance, knowing that some examples are better than others for introducing a concept encourages one to look for such a "start-up" example to introduce a new concept. Knowing about the knowledge encourages one to tell others about it. It is easier to share one's knowledge if one has a way of talking about it. It also reminds one to tell students important learning and understanding heuristics that one often forgets to mention because they are so thoroughly assimilated: for instance, instead of hoping that the students will come to recognize an example as a standard, why not tell them it is a "reference", which they should think of as an old friend to be used to test out conjectures and new definitions.

Thus, a conceptual framework for knowledge in complex domains can help organize known knowledge and direct exploration of new. By knowing about the knowing, one can be more expert in learning and teaching.

References

Brachman, R. A Structural Paradigm for Representing Knowledge. Report No. 3605, Bolt Beranek and Newman Inc., Cambridge, Mass. 1978.

_____, "On the Epistemological Status of Semantic Nets" in Associative Newtorks: Representation and Use of Knowledge by Computers. N. V. Findler (ed), Academic Press, 1977.

Bruner, J. S., Beyond the Information Given. George Allen and Unwin, Great Britain, 1973.

Buchanan, B. G., and T. E. Hendrick, Some Speculation about Artificial Intelligence and Legal Reasoning. Memo AIM 123, Computer Science Department, Stanford University, Stanford, CA., 1970.

Collins, A. and A. Stevens , Goals and Strategies of Effective Teachers Bolt Beranek and Newman, Inc., Cambridge, Mass., 1979

Davis, R., Detailed Description of Mathematical Behaviors that Demonstrate Understanding. Proposal to joint NIE/NSF Program, 1979.

Davis, R. B., Jockusch, E., and C. McKnight. "Cognitive Processes in Learning Algebra". The Journal of Children's Mathematical Behavior, Vol. 2, No. 1, Curriculum Laboratory, University of Illinois at Urbana-Champaign, 1978.

Davis, R. B., and C. C. McKnight, The Conceptualization of Mathematics Learning As a Foundation of Improved Measurement. Development Report No. 4 (Final Report NE-W NIE/G 76 0085), The Curriculum Laboratory, College of Education, University of Illinois, Urbana, 1979.

Dolciani, M. P., W. Wooton, E. F. Beckenbach and S. Sharron, Algebra 2, Houghton Mifflin Co., 1968.

Dunford, N., and J. T. Schwartz, Linear Operators Part 2: Spectral Theory. Interscience, New York, 1958.

Feigenbaum, E. A., and A. Barr (eds.), The Handbook of Artificial Intelligence. Wm. Kaufman, Inc., CA, 1981.

Gelbaum, B. R., and J. Olmstead, Counterexamples in Analysis. Holden-Day, California, 1964.

Greeno, J. G. "Understanding and Procedural Knowledge in Mathematics Instruction", Educational Psychlogy, Vol 12, No. 3, 1978.

Jacobs, H. R., Geometry. W. H. Freeman, San Francisco, 1974.

Kuhn, T. S., The Structure of Scientific Revolutions, Second Edition, Chicago University Press, Chicago, 1970.

Lenat, D. B., AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. Stanford A. I. Lab Memo AIM-286, Stanford University, 1976.

Lakatos, I., Proofs and Refutations, British Journal for the Philosophy of Science, Vol 19, May 1963, No. 3. Also published by Cambridge University Press, London, 1976.

Lindsay, P. H., and D. A. Norman, Human Information Processsing. Academic Press, New York, 1972.

Minsky, M. L., "A Framework for Representing Knowledge" in [Winston 1975].

____, Semantic Information Processing. The MIT Press, Massachusetts, 1968.

Myers, J. L., "Structure and Process in Learning Probability". Proposal to the NIE, Department of Psychology, University of Massachusetts, Amherst, MA., 1980.

Newell, A., and H. A. Simon, Human Problem Solving. Prentice-Hall, Inc., New Jersey, 1972.

Papert, S. A., Mindstorms Basic Books, New York, 1980.

_____, "Teaching Children Thinking" LOGO Memo No. 2, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass., 1971.

Polya, G., How To Solve It. Second Edition, Princeton University Press, New Jersey, 1973.

____, Mathematics and Plausible Reasoning, Volumes I and II. Second Edition, Princeton University Press, 1968.

_____ Mathematical Discovery. Vol. 2., John Wiley and Sons, Inc., New York, 1965.

Polya, G., and G. Szego, Problems and Theorems in Analysis I. Springer-Verlag, New York, 1972.

Quillian, M. R., "Semantic Memory" in [Minsky 1968].

Rin, H., An Investigation into Students' Understanding of Mathematical Definitions. Ph. D. Thesis Proposal, SESAME Group, University of California at Berkeley, Berkeley, CA., 1979.

Rissland, E. L., Example Generation. Proceedings Third National Conference of the Canadian Society for Computational Studies of Intelligence, Victoria, B.C., May 1980.

_____, The Structure of Mathematical Knowledge. Technical Report No. 472, MIT Artificial Intelligence Laboratory, 1978a.

_____, "Understanding Understanding Mathematics". Cognitive Science, Vol. 2, No. 4, 1978. Also published as A. I. Memo No. 488, MIT Artificial Intelligence Laboratory, 1978b.

Royden, H. L., Real Analysis. Second Edition, Macmillan, New York, 1963.

Rudin, W., Principles of Mathematical Analysis. Second edition, McGraw-Hill, New York, 1964.

Rumelhart, D. E. and Ortony, A. "Representation of Knowledge". In R. C. Anderson et al (Eds), Schooling and the Acquisition of Knowledge. Lawrence Erlbaum Associates, New Jersey, 1977.

Schank, R. C., and R. P. Abelson, Scripts, Plans, Goals and Understanding. Lawrence Erlbaum, Hillsdale, NJ., 1977.

Schank, R. C., and K. M. Colby, Computer Models of Thought and Language, W. H. Freeman, San Francisco, 1973.

Seely-Brown, J., A. Collins, and G. Harris, Artificial Intelligence and Learning Strategies Report No. 3634, Bolt Beranek and Newman Inc., Massachusetts, 1977.

Seely-Brown and R. R. Burton, Diagnostic Models for Procedural Bugs in Basic Mathematical Skils. Report No. 3669, Bolt Beranek and Newman Inc., Cambridge, Mass., 1977.

Steen, L., and J. Seebach, Counterexamples in Topology. Holt, Rinehart and Winston, New York, 1970.

Wegner, P., Programming with ADA: An Introduction by Means of Graduated Examples. Prentice-Hall, New Jersey, 1980.

_____, Proposal to the NSF, December 1980.

Winston, P. H., Artificial Intelligence. Addison-Wesley, Massachusetts, 1977.

___, The Psychology of Computer Vision. McGraw-Hill, New York, 1975a.

____, "Learning Structural Descriptions from Examples" in [Winston 1975a], 1975.

APPENDIX 3

TEACHING LEARNING SYSTEMS

by Sally Waisbrot

COINS Master's Project

Spring, 1981

Approved as to content and style by:

_____

Edwina L. Rissland, First Reader

_____

Elliot M. Soloway, Second Reader

Abstract

The author examines nine learning systems. The roles of the teacher, the training sequence, and primitive knowledge in these systems are compared and contrasted. Finally, the author discusses automating the teaching process and proposes a model for teaching systems.

TABLE OF CONTENTS

0. Introduction

1. Comparison of the Learning Systems
   1.1 Counting
   1.2 Constrained Example Generation
   1.3 Learning Structural Descriptions from Examples
   1.4 Hacker
   1.5 Some Studies in Machine Learning Using the Game of Checkers
   1.6 Learning=Interpretation+Generalization
   1.7 AM
   1.8 Pandemonium
   1.9 The Simulation of Verbal Learning Behavior

2. Common Features of Learning Systems
   2.1 Types of Teachers
   2.2 The Training Sequence
      2.2.1 Building Systems
      2.2.2 Refining Systems
      2.2.3 Discovering Systems
   2.3 Primitives and Piagetian Corollaries

3. Teaching Systems
   3.1 The Teacher
      3.1.1 Knowledge Needed
      3.1.2 Choosing the Training Sequence
      3.1.3 Evaluation
   3.2 The Model
      3.2.1 World Model
      3.2.2 Instance Group
      3.2.3 Critic
      3.2.4 Learning Element
      3.2.5 Summary

4. Conclusion

5. Appendix
   5.1 The Model
   5.2 Learning Systems
   5.3 Teaching Systems

6. Bibliography

0. INTRODUCTION

In the thirty year history of computer science, a large segment of time and energy has been spent in trying to create intelligent machines. However, what counts as intelligence is difficult to quantify. There is general agreement that the Turing test for intelligence is too stringent but beyond that, there is no way of assessing the intelligence of a system. Expert systems have been created that perform powerful tasks, but unless they have the ability to be modified they remain static in what they know and can do. On the other hand, if a system learns, that is, if it gains knowledge or it solves more difficult problems than it was initially instructed how to, then the system is intelligent and powerful. Thus, the ability to learn is central to intellegence, as I see it.

I have always been interested in learning and how one should teach in order to create the optimal environment for learning. In my six years of teaching young people, I observed how my style and presentation effected the success of the learner. I find that in computer science I am drawn to these same topics.

Every learning system, person or program, needs a teacher, and how dominant a role that teacher takes determines how the system learns. The most strongly directed systems use an active human trainer. This trainer has determined in advance how the learning

..., i.e., the order in which tasks will be executed ...size of each learning step. The trainer takes ...ponsibility for evaluating the performance of the system and ...king changes to improve the performance. At the other extreme ...systems which are completely self-directed. These systems use an "explore and discover" technique. They must be able to evaluate the quality or importance of their discoveries so that they can focus on exciting areas and reject dead ends. These two extremes are on a continuum of teacher guidance. In the middle are systems which are given some amount of training but provide their own evaluation, or train themselves but need careful feedback.

All the systems, regardless of how much direction they receive from the teacher, have some sort of training sequence. The strongly directed systems can have a predetermined sequence of examples or subtasks which have been planned by the author of the system. Sometimes, the teacher uses a trial and error approach to find the best training sequence, and feedback from the system helps the teacher choose better training examples. In more self-directed systems, a "correct" response is not known so the training sequence is planned to provide a variety of experiences which the system uses as practice and then evaluates by comparing its performance to other practice sessions. Completely self-directed systems must use some sort of agenda or heuristics for ordering tasks plus a comparison method for evaluating results.

The amount of knowledge and heuristics imbedded in a learning system effects how it learns. In general, systems which teach themselves must have a proportionately greater amount of knowledge and heuristics to give them the skills a teacher would use. Strongly directed systems, in general, have a simple knowledge-base, and use many of Piaget's elementary approaches to problem solving.

I examine these aspects of learning systems with the goal of creating a model of the teacher. Smith et al have created a model which unifies the characteristecs of learning systems so that they can be compared. I isolated the common attributes of teachers, not only so that teaching systems could be compared, but also so that they would mirror the learning system and the role of teacher and learner could be compared.

In this paper I discuss the characteristics of systems which learn from the perspective of how they are taught. In chapter 1, I give a general overview of the systems I analyze. In chapter 2, I focus on how the systems learn by examining the role of the teacher, the type of training sequence used, and the power of the initial knowledge and heuristics given the system. In the final chapter I discuss the common features of the teachers of the systems and propose a model for a teaching system.

1. COMPARISON OF THE LEARNING SYSTEMS

Since the bulk of this paper concerns details of nine learning systems, I would like to begin by familiarizing the reader with these programs. In this chapter I will briefly summarize and compare the systems. The examples cited serve not only to clarify the procedures and goals of these systems, but also to provide a reference point for further discussion.

Smith et al [Smith, 1977] have formulated a model for detailing functional components of learning systems. Every system uses different terminology and perspectives so using this framework unites the discussion and allows comparison. An explanation of the facets of Smith's framework, plus the representation of each of the learning systems discussed is located in the appendix.

1.1 Counting

by Oliver G. Selfridge

In Counting, the goal is to make the system count the number of objects on a table. To aid in the process, there is a box with a number on it which can be incremented or decremented, and a pointer which moves left or right across the objects on the table. These four processes are called primitive capabilities (PCs).

PC1 Left

PC2 Right

PC3 Increment

PC4 Decrement

Initially, the user cannot instruct the system to count because the task is too difficult with the given primitives. (since Count will merely combine progressively longer sequences of its PCs until it stumbles across the correct combination which produces the count, or uses up a prescribed amount of time.) The key to getting an accurate count procedure, is to divide that process into subtasks which <u>are</u> easy to achieve by Count's trial and error style. One sample subtask would be: zero the number in the box. This would be accomplished by repeated decrements of the counter (a string of PC4s), something entirely within the realm of possibility for a trial and error process. This process is named and becomes a "derived capability" (DC), and is added to the list of possible components of a new task given the system.

In empirical studies, the author found the following sequence of subtasks to be of the proper size to enable the system to learn to count:

User defined subtask	Count's reaction
DC1. Zero counter	series of PC4
DC2. Move pointer all the way to left	series of PC1
DC3. Increment counter, move right	PC3,PC2
DC4. Count	DC1, DC2, series of DC3

Success! The system counts Of course this is not the only solution. The order of DC 1-3 is irrelevant. Also the pointer could as easily have been pushed to the right as left.

1.2 Constrained Example Generation

by Edwina L. Rissland and Elliot M. Soloway

Like Counting, CEG builds on its repetoire of abilities by adding the new knowledge it learns to its examples-space. CEG begins with a small set of primitive examples. Its goal is to modify these primitives to create a new enlarged set of useful examples. Again, the user acts as teacher and his/her choices play a large role in determining success and efficiency of the program.

The user chooses a domain: an initial examples-space and few modification routines which drive the learning process. The user then presents the system with problems much as Counting is presented with subtasks. These problems have a certain number of constraints which must be met. A retrieval mechanism checks each member of examples-space against the new problem and a judge records how many constraints are met. If all constraints are met, then the problem is in examples-space, a retrieval is done, and the system has succeeded. Otherwise, the system attempts to modify examples from examples-space to solve the problem. Since it has a scoring of how many constraints were met by each example during the retrieval process, the members of examples-space are now ranked, indicating the likelihood of success by the

modification routines. Now, each candidate, in order, is presented to the modification routine and constraint satisfaction is again tested. If none of the modification techniques is successful on any of the candidates, a new example must be generated from scratch. This aspect of the system has not yet been implemented and so I will confine my discussion to the retrieval and modification portion of the program.

An example from everyday life is as follows: You know one route which you walk through the city to the store. (your examples-space) On a particular day, as you are walking along this path you see a particularly feroious looking dog in the next block whom you would rather not confront. (You want to modify your route.) You know that if you go left, right, right, left, at the next intersections, you will end up one block ahead of where you are now. (modification technique) You try this and it works, so now you have a new route to the store which avoids the ferocious dog. (addition to examples-space)

From the linear domain, a typical problem would be: Find the equation of a line through (1,2) and (4,8). (Note how the program relies on trial and error as Count did.)

Examples-space: y=2

y=x

Modification techniques: Function times const

1. Check to see if a solution to this problem is in examples-space.

a. y=2.

The first ordered pair works, but the second does not.

Score=1.

b. y=x. Doesn't work at all. Score=0.

2. Try modification

a. Since y=2 got a score of 1, modify it first.

The modification routine chooses const=4 => y=2(4).

Still this works only once. Score=1.

b. The modification routine chooses const=2, so

modify y=x => y=x(2). This works. Score=2

This example, y=2x can be added to examples-space.

1.3 Learning Structural Descriptions from Examples

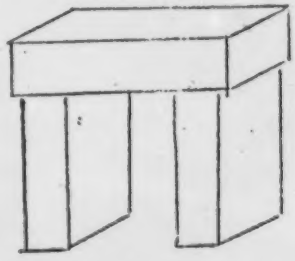
by Patrick H. Winston

The key idea in Winston's system is "near miss". A near miss is a representation which is one step away from correctly representing a concept, and it plays the role of counterexample or refutation in learning the concept.

In this system, a structural description of a concept in Blocks World is built up from a combination of examples and near misses. The first example presented to the system causes it to build the initial network description. Next, a near miss is presented. The system compares this with its network and finds the one place that they differ. Now the network must be modified to include the new information learned. As in CEG, this system has modification

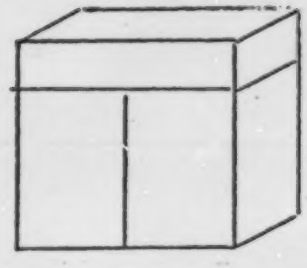
techniques to try, but they are more numerous (21) and instead of being tried serially until one fits, each near miss situation triggers a specific modification rule. For example, if an example has one element characterized as a brick, and a near miss has a wedge in the same position, the description is changed by adding a "must be" pointer to the brick node. The system learns from examples as well. In the previous situation, if both had been classified as examples, then the representation would need to accomodate both a brick and a wedge. A "merge" would take place, and the word object would replace the word brick. After a training sequence of examples and near misses, a generalized version of the concept is built which can identify whether new examples fit the concept or not.

The classic example is the blocks world arch. Since I am interested in the training sequence more than the representation, I will use a simple English description of what is happening, rather than showing Winston's networks.



EXAMPLE

1. Brick A and brick B support brick C.

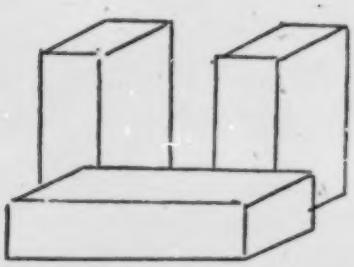


NEAR MISS

2. Brick A and brick B support brick C, but A must not touch B.

EXAMPLE

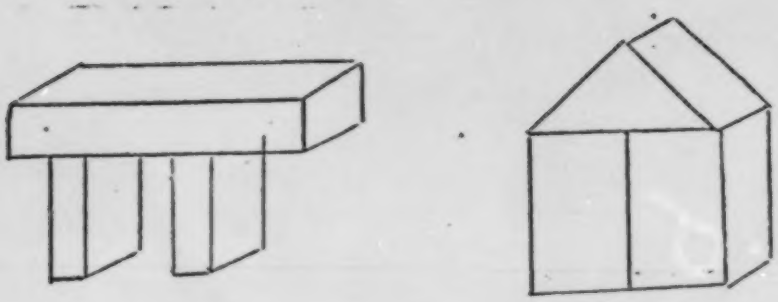
3. Brick A and Brick B support object C, but A must not touch B.



NEAR MISS

4. Brick A and Brick B must support object C, but A does not touch B

Now if the newly trained system is presented with



the first would be accepted as an arch and the second rejected.

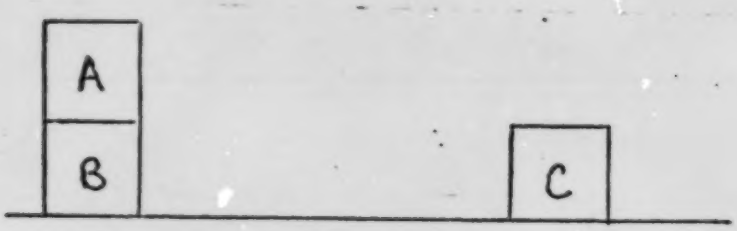
1.4 Hacker

by Gerald J. Sussman

Hacker is another blocks world system which creates programs for building increasingly more complex towers. Hacker has the primitive ability to put one block on another. When Hacker is given a problem which cannot immediately be solved by this put-on program, it modifies that program and adds it to its program library, just as CEG adds examples to examples-space. Modification heuristics for this system are far more complex than any of the others studied this far. Hacker takes a program from its library which seems applicable to the problem at hand and tries to decide why it doesn't work. Sussman has classified all six types of bugs he expects to appear in programs and Hacker can pick the proper bug and and find the corresponding proper patch which is also available in the heuristics. The program is then tested to insure that in solving problems, it has not created new ones. If this type of program arises again, it will now be in the answer library and can merely be retrieved.

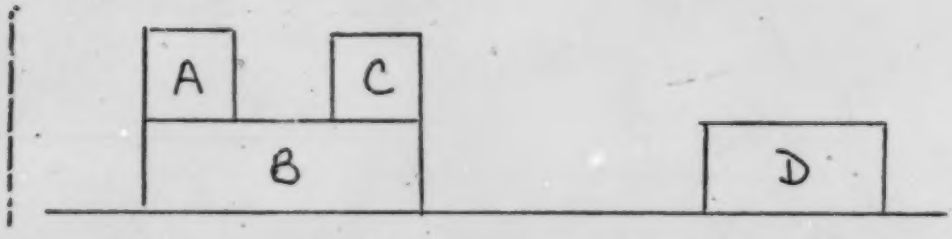
Two examples, one simple, the other complex, illustrate how Hacker works.

Problem 1: (MAKE (ON B C))



The answer library contains a program for putting B on C but the bug classifier complains that there is an "unsatisfied-prerequisite". In other words, since Hacker can only move one block at a time, the program to put B on C requires B to have a cleared top. The Blocks World Knowledge Library defines what "cleartop" is and the Programmming Techniques Library provides the heuristics for adding lines to the put-on program which removes blocks one at a time until B has a clear top.

Problem 2: (PROTECT (ON C B))
(MAKE (ON D B))



When Hacker reaches this point, its Answer Library has expanded. It has a compaction program which seems like a good approach. It moves all blocks left as far as possible. After moving C left, there is still no room for D and the blocks cannot be compacted

further. The other choice is to "unclutter" B. (There is a program for this.) C is protected so the only uncluttering that can be done is to remove A. Hacker, however will not permit this since it will undo the compactness of the top; "strategy-clobbers-brother," Hacker complains. Now Hacker tries uncluttering (i.e., removing A) first and D fits. Cleverly, the Program Critic is now instructed that when uncluttering and compacting are both potential solutions to a problem, try uncluttering first.

1.5 Some Studies in Machine Learning Using the Game of Checkers

by A.L. Samuel

In fewer than 30 games against "the book", checker masters, and itself, Samuel's program uses experience to learn to play a better than average game of checkers. It uses strict numerical computation in making its decisions rather than using domain specific heuristics, however, the things being measured by the numbers, such as control-of-center, do embody domain knowledge.

The program knows how to play checkers, that is, it knows legal moves. It creates a lookahead tree of possible moves. Since there are 10**40 possible moves in checkers it would not be sensible to lookahead to the end of the game. Therefore, a 20-ply lookahead limit is set with the option to look farther in interesting circumstances.

The value of a particular move is determined by 38 parameters which highlight the various aspects of the game. Foremost among these parameters is piece advantage since it is assumed that it is always advantageous to have more pieces than one's opponent. This parameter is always present in the decision of the next move. Another parameter whose value may vary is "denial of occupancy" where the total score is increased for each square where an opponent may be captured in a single move without exchange. The next move made is the one with the highest score, taking into account the fact that on every other move the opponent makes the optimal move as well. One disadvantage of this strategy is that the checker player is confused by poor moves by its opponent.

The program improves its game by altering which parameters contribute to the score and changing the coefficients of these parameters. The two modes it has for doing this are called rote learning and learning by generalization.

In rote learning, Samuel chose the linear polynomial (using only four parameters) which forms the score of each move. During play, all board positions were saved so that the checker player could repeat good moves and delete bad ones. Redundancies and little used positions were culled.

Learning by generalization is more interesting because the system chooses its own parameters and weighting coefficients. Initially,

16 parameters are chosen at random. The difficult decision is how often to change the parameters and coefficients. Too often would cause the quality of the game to fluctuate wildly, yet too rarely would slow the learning. Remember, this theme from Counting as well. Samuel chooses to have the machine play itself. The Alpha player changes its polynomial after each move, based on a comparison with the stored rote learning board positions. Beta keeps the same polynomial for an entire game, but adopts Alpha's polynomial when Alpha wins. If Beta wins three times, however, Alpha's polynomial is changed radically by making its leading coefficient zero.

Rote learning was most successful in the opening and end games while learning by generalization played a good middle game. This shows much hope for combining the two methods.

1.6 Learning=Interpretation+Generalization

by Elliot M. Soloway

Soloway's program learns the rules of baseball by observing a nine inning game. The important distinctions between this system and those previous discussed are that it is knowledge-directed and the processing is multileveled.

To understand just how knowledge-directed the system is, contrast the 3300 features observable in one baseball game with the few

features present in representing Winston's arch. Instead of explicitly being told what features to watch for, Soloway's program has this information in the form of heuristics. These heuristics form the multileveling. On a low level, the program observes a "throw", or a "swinghit", or a "catch". On a higher level, it recognizes a "ground out", and on the highest level it groups the class of "outs". More specifically, there are nine levels of pattern description: the first three highlight important features and discard irrelevant ones, the next three generate hypotheses about these actions based on facts about competitive action-oriented games, and the last three create generalized classes. The system evaluates itself by making predictions about events and increasing confidence in a theory when a prediction holds true.

An example in more detail will illustrate the multilevel interpretation. At the lowest level, each snapshot has 18 features (runs, hits, outs, position of each player), but if several snapshots are observed together, most features remain constant and are filtered out. This shows the power of the heuristic "attend to change". Suppose the only active features remaining are: pitcher throws ball, then pitcher stands still, and batter stands still, swings, hits. We now have an episode: an area of high activity surrounded by low activity. Moving to the next level, facts about competitive games allow the system to infer that the pitcher's throw enabled the batter's hit and because the pitcher's fast throw and the batter's hit are difficult to execute, then these two players must be opponents.

Examining several episodes, the system would discover a ground out at first base. After viewing a large portion of the game, having generalized ground out, fly out, double play, the system creates the class "out" and an important aspect of the game of baseball has been discovered.

1.7 AM

by Douglas B. Lenat

Lenat introduces the interesting idea of learning as a discovery process. While some systems discover relationships among concepts and others define new concepts, AM attemps to do both and without the aid of a teacher.

The knowledge base is huge and even Lenat hesitates at this at this as he asks "Is the knowledge base finely tuned to elicit this one chain of behaviors?"[Lenat, 1976, p18] AM starts with 100 partially defined concepts and 250 heuristic rules. The concepts are represented as a bundle of slots, i.e., a frame, such as interestingness, analogies, algorithms. An agenda uses the heuristics to decide if a facet of a concept is worth working on and supplies it with a priority rating. The heuristics are condition/action rules. Each heuristic is attached to the most general concept for which it applies. The current task flips back through the generalization links to find all relevant heuristics. These heuristics are then executed, i.e., a new task is added to the agenda, or a new concept is defined, or a facet of a

particular concept is filled. In a one hour run, AM discovered 300 concepts, one fourth of which were considered by the author and AM to be significant.

One of the more interesting discoveries of AM was the concept of primes which it found in the space of three tasks. A task looks like this:

Task: Fill in Examples of the concept "Divisors-of".

3 reasons:

(1)No known examples of Divisors-of yet.

(2)TIMES (related to Divisors-of) is now very interesting.

(3)Focus of attention: AM just defined Divisors-of.

26 examples found, in 9 secs. e.g., Divisors-of(6)={1,2,3,6}

The next task is to consider numbers having small sets of divisors since AM just worked on divisors and knows the heuristic that extreme cases are worthwhile. It discovers no numbers with 0 divisors, one number with 1 divisor, no conjecture for numbers with 2 divisors but they merit further study, and numbers with 3 divisors are all perfect squares. In the final task of this series, AM considers numbers with three divisors since they were just defined and proved unexpectedly to be perfect squares, and discovers that their square roots are numbers with two divisors. Thus the "interestingness" of these numbers is boosted and the user then names these numbers primes.

1.8 Pandemonium

by Oliver G. Selfridge

Pandemonium is a system designed to distinguish the dots and dashes of a morse code transmission, or more generally to recognize patterns that have not been specified. It stands as an early representative of attempts at machine learning.

The organization of the system has four levels composed of different types of demons. At the lowest level, data demons store and pass on data. Computational demons perform computations on the data. Third level, cognitive demons weigh the evidence and create a score, while the super-demon selects the demon with the top score or "loudest shriek". The adaptive learning takes place at the computational and cognitive demon level. The cognitive demons use a hill climbing technique to alter the weights of the subdemons. The make up of computation demons is altered by removing those which prove unproductive and mutating the survivors. This process is meant to produce an evolution of the demons, a survival of the fittest.

An example from the morse code domain may clarify the roles of the various demons. The data demons receive a sequence of duration lengths, alternately marks (i.e., dots and dashes) and spaces. The computational demons check attributes such as: Are two durations of equal length? How does this compare with the maximum duration so far? To what degree has the ith duration been catagorized as a dot or dash? The two cognitive demons, dot and dash, compute the weighted sum of 150 computational demons. The

decision demon chooses dot or dash based on which cognitive demon has the highest score.

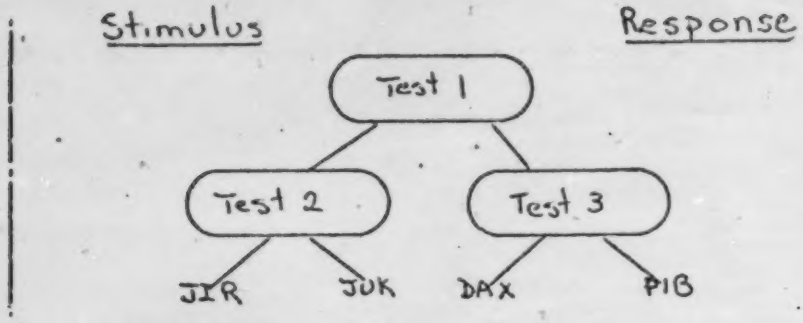
1.9 The Simulation of Verbal Learning Behavior

by Edward A. Feigenbaum

EPAM (Elementary Perceiver And Memorizer) is the only system studied which models the human learning behaviors of interference and forgetting. EPAM simulates the psychological experiment where a subject learns to associate pairs of nonsense syllables.

As each stimulus-response pair of syllables is presented, EPAM builds a discrimination net so the response can be retrieved. The net is made up of tests whose results indicate which branch of the net to follow. The tests are based purely on visual cues from the syllables; there is no auditory discrimination, or discrimination based on association.

The net is built slowly so that a test which is adequate at one point can cause errors later. For example, when the pair JIR-DAX is given, a test for the first letter J is sufficient to identify the stimulus. But when the pair JUK-PIB is added, the letter J is ambiguous and a further test must be created.



Information can be temporarily lost or "forgotten" in a growing network. Lost information can be restored by adding more cue information.

2. COMMON FEATURES OF LEARNING SYSTEMS

2.1 Types of Teachers

The behavior of a learning system is strongly influenced by the way in which it is taught. Teaching styles range from completely teacher directed to the situation in which requests for instruction are initiated by the learner.

The extreme case would be exemplified by a system such as EPAM or Winston, in which the teacher simply feeds information into the learning system. The human learner, who after listening to a lecture can spew back the contents, or the child who memorizes the multiplication facts, is said to have learned. This is because all human learning involves an element of forgetting and one is given credit for overcoming the obstacle of forgetfulness. Machines, on the other hand are expected to show a degree of interpretation or generalization before they are considered to have learned. One exception to this is Feigenbaum's EPAM, which learns pairs of nonsense syllables. The trainer gives all the information the system needs to know, but the system, instead of recording the pairs verbatim, stores only enough cues to make the correct decision at that point in time. As the example in the previous section showed, at the addition of the stimulus JUK, the test for first letter was not enough to discriminate between JIR and JUK. An error was made and the tests revised. Since EPAM attempts to mimic human behavior, making errors when syllables are similar, showing interference when new syllables are introduced,

or forgetting pairs after knowing them for a while, the forming of these memory associations is considered to be learning.

At the other end of the spectrum are two systems which are learner directed: Soloway's baseball program and Lenat's AM. Both are knowledge-directed, that is, initially, all relevant knowledge is imbedded in the system rather than being presented in some incremental way by a teacher. The system then procedes independently to discover rules, in Soloway's case, and elementary mathematics concepts in AM. An easy criticism to make is that with so much knowledge and heuristics given, how could the systems help but learn what their creators intended them to learn. Or how can one fail to point out that the amount learned is miniscule in comparison with the imbedded knowledge? Soloway points out [Soloway and Riseman, 1977, p. 181] when faced with the huge number of features present in any concept learning system, domain knowledge is required to limit search; Lenat too uses the theme of heuristic search. The knowledge used in these systems corresponds to the knowledge a teacher needs to classify and order training sequences. To illustrate, recall the arch training sequence which Winston uses. While Winston hands his system one example or near miss at a time, Soloway provides his system with the entire group of snapshots at once. Far from making this easier for the system, it now has to do some of the work Winston did as trainer: Soloway's system must group snapshots together to form an episode before he has the amount of information present in one example or near miss. Lenat's system has an even more difficult task. It, too, has all the concepts it needs, but it

has to generate the worth ratings which determine the order in which tasks are undertaken. No one told AM that to discover primes it should examine the concept divisors of and then look at extreme cases. Lenat had to wait until the worth rating was high enough that AM chose to examine it. The fact that there were three times more uninteresting concepts discovered as interesting ones attests to how slow a process this was.

Between the two extremes are the remaining systems which to a more or less structured degree are guided by the trainer. Winston's program which learns concepts through examples and near misses is the most highly structured of the systems I am discussing, that is, its success is dependent on the quality of the training sequence. The choice of a good initial example/near miss pair is essential: If the example is too specific: "two bricks supporting a wedge", then the near miss must be equally specific: "two bricks not supporting a wedge". On the other hand, Winston need not be concerned about presenting too general an example, as in "an arch is an object supported by two objects", because his blocks world descriptors always choose to call the blocks by their specific names (Brick A and brick B support brick C.) even though they are hierarchically linked to the general term "object". This is a subtle heuristic which gives his system the ability to make a good start. Imagine starting with a near miss! The system would have to build a representation without knowing where it went wrong. This may work, but it would procede much more slowly. Suppose the arch example were run in reverse, starting with the disassembled arch. While the disassembled arch is one step away

from being an example, it is two steps from matching the particular example with a wedge on top. The system would not be able to tell if the important aspect were that the third object must be a wedge or must be supported. If it decided in favor of a wedge, its decision would be confirmed by the near miss with the touching sides, since its third object is not a wedge. Only when it reached what was once the initial example (small wonder why it was chosen as the initial example) would it discover its error.

Rissland's CEG and Selfridge's Pandemonium start with the need for a lot of guidance but require less as the system learns more. In CEG, initially, examples-space and the modification techniques are the only knowledge the system has. But after several sessions, examples-space has grown and CEG can create new examples from its own examples. It would be even more exciting if it could create modification techniques from its own modification techniques, and perhaps that will be the goal of the constructing phase of CEG. Pandemonium's training is in the form of feedback. After each decision, the trainer tells the system whether it is right or wrong and it adapts accordingly. After a while, it is assumed, the computational demons will be finely tuned enough that one of the two cognitive demons will always outshine the other. At this point the system will be able to train itself to choose even better computational rules forcing the dot and dash demons to be even more distinct. One can see that the structure from the teacher decreases as the system's domain knowledge increases.

Hacker is an exception, because despite its answer library, programming techniques library, and critics gallery, it still requires careful direction from the teacher, so that its "lessons" do not come in chunks too large to handle. (Counting, too, was concerned with this problem.) For example, in the two previously cited examples, if the cleartop program of the first, simpler problem had not been written, as well as intermediate programs for compacting and uncluttering, then the complex second problem couldn't have been solved. I often wondered how Hacker would have reacted to an impossible problem. For example, what if the second problem had been: (PROTECT (ON C D)) (PROTECT (ON A B)) (MAKE (ON D B))? I believe Hacker would have churned away in the programming techniques library hoping vainly to find that last bug.

The trainer of the checker player differs from the other trainers because s/he has no preconceived "right" game as Winston has a correct representation of an arch or Soloway has proper rules of baseball. The trainer acting as opponent has a goal to provide the system with a lot of experience playing checkers and then at the end, decide if there has been improvement.

In Counting, the roles of trainer and learner in many ways were indistinguishable. The human trainer suggested goals (but no methods) which the system tried to achieve, for example, "Move pointer all the way to the left". The system on the other hand provided feedback to the trainer when a goal was too difficult, thereby teaching the trainer how to choose better problems.

It is apparent that there is a direct tradeoff between amount of structure imposed from a teacher and amount of knowledge needed within the system. It is useful to consider all systems to be knowledge directed. Those with strongly directive teachers simply received their knowledge incrementally while the rest need less guidance since the training knowledge is provided within the system.

2.2 The Training Sequence

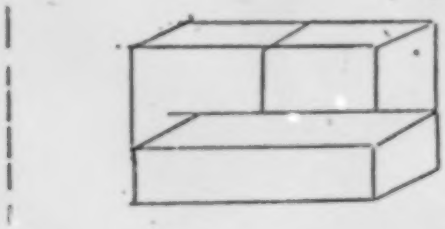
Closely related to how much structure the teacher imposes on a learning system is the training sequence used. Those systems which require careful guidance from the teacher were example driven. I call this group building systems, since they started with their primitive capabilities and through a sequence of examples, built a more and more complex concept, scene, or skill. Other systems, less strongly guided, learned by a series of examples which had no particular ordering. These could be considered to be practice driven and I call this group refining systems since they improved upon a skill rather than building one. The last group, the learner directed systems, were knowledge driven. I call them discovery systems, because they used their own heuristics on the knowledge they had to discover rules or concepts.

^&Building Systems	Refining Systems	Discovering Systems\&
Winston's	Pandemonium	Baseball
Hacker	Checker Player	AM
CEG		
Counting		

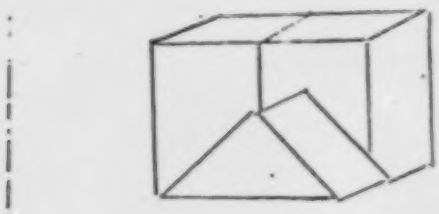
2.2.1 Building Systems

Each of the four building systems: Winston's, Hacker, CEG, and Counting, uses a different method of increasing its knowledge. Winston's concepts grow from a sequence of examples and near misses which differ from the example in just one aspect. Is the key to learning the nearness of the miss? If a near miss were presented which differed from the example in two aspects then the

system would not know which aspect to recognize. It is possible that the confusion lies not in the fact that there are two differences, but that the system expects only one. A funtion NEAR(2) MISS (or NEAR(3) MISS or NEAR(10) MISS)would inform the system that the new example was so many differences away. This would result in faster learning. For example



is two steps away from being an arch. If the system expected only a near miss, then it would have no way of figuring out whether support or touching was important, but if it were instructed to find two deviations from the example, there would be no problem. However, Sussman points out that a major difficulty in learning is the unexpected interaction of competing subgoals. Somewhere, a "miss" will undo another miss and the system will not be able to understand. For example



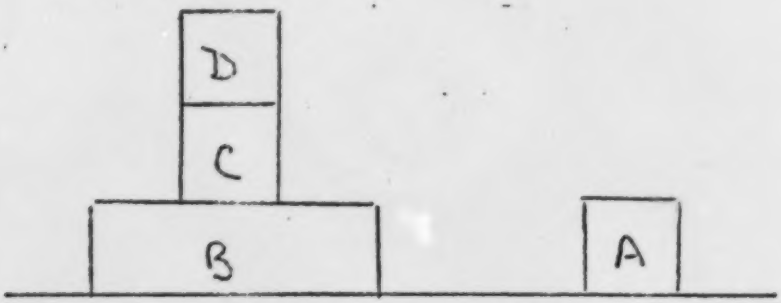
is also two steps away from being an arch. But if the example with a wedge on top has not been presented, then the system sees three potential misses: wedge, touch, and supported by. It can't resolve the problem. The beauty of Winston's choice of single near misses is that the order of presentation of examples is made

less important. As long as the initial example/near miss sets up a general enough representation, the system would work equally well if the five examples were followed by five near misses. This was the only system in which the initial examples held such a unique role.

Hacker develops a skill at tower building by solving progressively more difficult problems. Sussman's training sequence in each scenario follows Winston's method of building knowledge by moving one step away from the previous example. That is, he begins each scenario with a problem that the system can solve then follows with a problem which requires only one patch from the programming techniques or debugger. After the two problems mentioned Hacker is asked to solve:

Problem 3: (PROTECT (ON C B))

(MAKE (ON A B))



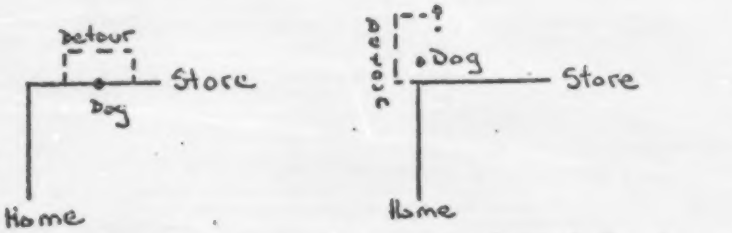
This problem requires it to combine the cleartop program of problem 1 with the compaction program of problem 2, forming a new "make room" program. Since Hacker builds on previous knowledge, it would slow the system considerably to take larger learning steps. In that situation it would be forced to choose which problem to solve first and it does not have the heuristics to do so efficiently. One way that it does speed up its learning is by

generalizing. In the cleartop example, rather than writing a program which takes one block off the top of another, Hacker keeps taking blocks off until the one needed is uncovered. (just as Counting repeatedly decrements the counter until it is zero. Then when a block which is under three others is to be moved, the original cleartop program suffices.

Selfridge's Counting is a system in which the user, not the creater acts as teacher. In a way very similar to Hacker, the teacher must break the goal of counting into subproblems beginning with a task the system knows how to solve and advancing to more difficult problems taking just the right sized steps. Again, too large a step results in a long, possibly too long search and too small a step is inefficient use of time. The difficulties in doing this are twofold. First, it is hard for people with programming experience not to try to tell the system <u>how</u> to do each subtask--and this is impossible. More importantly, it is difficult to choose the level of subtask. A subtask which is too large, such as "move the pointer to the right ten notches and increment the counter by ten (with a goal of counting objects ten at a time), will cause count to run out of time. On the other hand, the subtask sequence: (1) move pointer left 2 (2) move pointer left 3... is too slow a progression. Count will merely find a sequence of two PC1s, then this new DC and another PC1. The reason the user does not want to proceed at this careful pace is not only the intuitive feeling that it is a waste of time, but also the fact that the library of PCs and DCs will become so large that Count will reach its time limit while still working on short

strings of PCs and DCs, since it systematically searches through all possibilities.

In CEG, the user not only acts as the teacher, but also is responsible for choosing the initial examples-space. Problems are presented to be solved by a retrieval or modification mechanism. Again, an effort must be made to strain the modification routines just enough that examples-space will be expanded but not so much that the problem is unsolvable. In the real world example, you wouldn't want to use the modification routine just before a right hand turn, because it would leave you off your path and you would never find the store.



In the math domain, an example ordered pair (4,2), (4,7) would be unsolvable because a vertical line has undefined slope.

As can be seen in these four examples, the "one step away" policy for choosing problems is the strongest theme.

The four different building systems show four different methods of building knowledge: example and near miss, scenarios, division into subtasks, and modification. Why so many methods? It is

interesting to speculate how some of the systems would function using a different method. Winston's work is the least flexible since it has no feeling of subproblems. It also differs from the other three in acquiring a concept rather than a skill. Though Hacker works best with its scenario scheme, it shows similarities to both Winston's and Selfridge's work. Like Counting, breaking a problem into subtasks (for example, first unclutter, then compact, then put-on) plays an important role. And like Winston's system, Hacker has the feeling of near miss. Here, every problem is a near miss, needing just one more change. The Counting and CEG frameworks are interchangeable. To put Counting into the CEG framework, merely call the PCs, examples-space. The modification technique would be "combine PCs in increasingly longer strings". When the user supplied problems, the modification technique would create new examples for examples-space. If CEG used Counting's method, all of examples-space would be considered to be PCs. When a problem was presented, instead of combining PCs, the system would first check the PCs then each of the modifications on the PCs. A problem with this method is that trial and error is an inefficient search technique. Even a simple problem may be difficult to find because the proper modification is checked last.

2.2.2 Refining Systems

The refining systems, Pandemonium and the Checker Player, could be considered adaptive learning systems. Neither requires a training sequence because input is assumed to follow the real world rules of morse code transmission or checker games. In fact, the learning results from changes in parameters or weighting due to

feedback on performance. An analogy between these systems and the example driven system is the difference between making a mistake and being told how to correct it, and being instructed how to avoid the mistake in the first place [or in programming, the difference between a repeat loop, which checks a condition at the end of the loop, and a while loop, which checks it at the beginning]. Since much of Pandemonium's learning comes from hill climbing techniques, Selfridge hopes that the training examples "encourage the machine or organism to get enough on the foot-hills so that small changes in his parameters will produce noticeable improvement in his altitude or score". [Selfridge, 1959, p521]

There are corollaries between how these refining systems are trained and the sequence of examples given to the example driven systems. Just as examples must be presented to the systems in carefully incremented steps so must the parameters of the checker player and the weights of the demons be tuned with equal care. Too much responsiveness to the environment (i.e., changing parameters after each move or transmission) causes wild changes whereas, making changes rarely slows learning. Both Samuel and Selfridge found that a two-fold adjustment scheme, acting as a check and balance on each other, was most successful. Samuel uses hypothetical checker players Alpha and Beta. (Remember, Alpha changes its paramenters after every move while Beta keeps the same polynomial for the duration of the game. If Alpha wins the game, then Beta is given Alpha's polynomial.) So, Alpha takes wild risks but Beta tempers things. Selfridge relies on mutated fission (i.e., dropping low weighted demons and reweighting survivors),

and conjugation (i.e., creating new demons by performing binary functions on two useful demons). Conjugation can make fairly large changes, while mutated fission acts as a normalizing function.

Why aren't all learning systems of the building type? One could attempt to put Pandemonium in an example driven framework. Dots of a certain length would be near misses of dashes and vice versa, but this, I'm afraid would destroy the whole spirit of the shrieking demons. One might consider a training sequence of checker games from the simple minded to the expert but the play of the novice may be no simpler to analyze than that of the expert.

2.2.3 Discovering Systems

The discovering systems, Baseball and AM, while they don't have explicit training sequences use heuristics to train their systems. AM discovers concepts or tasks, or fills in facets under the direction of an agenda. Just as Hacker cannot build a complex tower without learning simpler ones first, AM cannot discover the concept "primes" without previously discovering "divisor", and before that, "times". In other words, order of learning is as important here as it was in the example driven systems. The agenda directs the order by choosing the task with the highest worth rating. This worth rating in turn is formed by summing the values of the reasons for exploring a new task. The result is that instead of building toward one goal, AM often builds toward

several goals at a time. To prevent too much jumping around, AM gives a large value to topics which were recently found interesting. This gives the feeling of speeding up learning. AM leaps from divisors to their extreme cases (numbers with 0, 1, 2, 3 divisors) to numbers with three divisors at which point primes come up again. Their worth is boosted, having twice been found interesting recently, and AM is off on an exploration of the properties of primes.

Heuristics in Soloway's system, too, provide an ordering of goals from simple to complex. Like Hacker's training sequence from simple to difficult problems, Soloway's training sequence could be considered a progression from low level snapshots to high level generalizations, with the system driving itself through the sequence. Soloway does not have an agenda mechanism as AM does, but uses his multilevel approach to decide which task comes next. The knowledge building systems are not really very different from the example driven. One can find their trainers encoded as rules within the system.

2.3 Primitives and Piagetian Corollaries

When learning systems come under attack it is usually the quantity and quality of their primitives which are criticised. The knowledge to be learned was imbedded in the system, critics claim.

Rosenberg, an advocate of Piagetian theory states,

> Current AI learning programs work by assuming some initial body of knowledge and learning techniques, and then showing how additional knowledge and techniques could be learned. However, this creates a regress; how the starting knowledge and techniques are acquired is left for someone else to explain. In addition, the initial knowledge used is often much more advanced than what is to be to be acquired; for example, Sussman's HACKER uses fairly sophisticated debugging techniques to learn something children acquire be age two. [Rosenberg, 1980, p 267]

When faced with a new problem, AI programs should

> Engage extensively in trial-and-error behavior
> Strive to find regularities in the outcomes
> Use the regularities to construct and the overgeneralize a theory
> Construct confirmation tests of the theory and note the exceptions
> Construct a theory for the exceptions, if they are numerous enough
> Try to coordinate the two theories into a larger one
> Test this resultant theory by confirmation and refutation tests. [Rosenberg, 1980, p 268]

In this section, I discuss how the primitives and learning techniques of a system effect what it learns, whether or not the primitives and learning techniques are more advanced than the learning, and if that matters.

Perhaps it is bacause EPAM was meant to simulate human behavior that it has the characteristics of a simple knowledge base and most closely follows the Piagetian problem solving techniques. EPAM knows how to build a network of binary tests on syllables.

The only tests are first letter, third letter, and second letter tests given in that order. It uses the regularities of its input to construct a theory (first letters will differ), finds this to be overly general when it meets an exception, and then reconstructs a theory for the exceptions (look at third letters).

It is impossible to resist citing Counting as an example of a system with very few and very simple primitives. Of course, it does start with an initial body of knowledge and learning techniques but these are kept to a minimum: a model world of a box, a table, a counter, a pointer, four PCs and the ability to combine them, using heuristics such as "repeat" (indefinitely) or "N times". Though the system learns to count, the PCs do not hint that this must be the task learned. The PCs "increment" and "decrement" make one think the domain is numerical, but the left-right pointer does not. In fact, Selfridge planned that the goal of the system would be learning to reverse a string of letters, which, though it is a far different procedure from assessing the number of objects in a box, and is fairly non-numerical, can use the same PCs. If one examines these PCs in a still different context they bear a strong resemblance to the operations permitted for a Turing machine. The learning technique which Counting is provided with is merely trial and error combination of PCs, starting with short strings and using some hill climbing to limit the search of longer and longer strings. I consider this a simple heuristic since it follows the primary Piagetian charateristic of problem solving in young children : engaging in trial-and-error behavior to construct a theory.

CEG can vary enormously depending on the choice of domain. Whereas Counting could learn several different skills using the same knowledge base, in CEG, the knowledge depends entirely on the domain and associated primitives. However, like Counting, the domain chosen by the authors has a very simple set of primitives and heuristics. In the CEG example cited, the one heuristic, "function times constant", allows examples-space to expand from a base of horizontal lines and a diagonal line through the origin, to a larger space containing all the sloping lines through the origin. Certainly, what has been learned is far more complex than the original heuristic which merely amounts to tipping the line this way and that until it matches up with the two given points.

Pandemonium and the checker player, the two adaptive learning systems, start with "how to" knowledge and learn "how to skillfully". The primitive knowledge is as simple as it can be: Samuel's program is meant to learn to play a good game of checkers so it must know legal moves; Pandemonium must recognize fuzzy dots and dashes so it must know about idealized dots, dashes, and spaces. The next level of built in knowledge is more complex. The checker player has 38 parameters dealing with 38 ways to judge the quality of a move. These are characteristics of good play that only an expert would know. Certainly, the checker player uses a simple trial and error (or better, trial and refinement) heuristic for choosing the best parameters but one would expect the novice checker player to <u>generate</u> these parameters rather than choose and weight them. The Pandemonium demons know tricky numerical functions which they apply using simple hill climbing

heuristics, but why were they given the functions rather than being forced to discover useful computations? The choice in these two systems was whether they did their jobs or did them at an expert level. The checker player's goal was to play an expert game of checkers, Pandemonium's to recognize dots and dashes with high reliability. Therefore, both authors gave them the tools which they needed to learn to use effectively in order to acquire their respective skills, instead of having the goal be generating the tools.

These five systems have used fairly simple knowledge and heuristics to achieve their goals, while the remaining ones use both sophisticated heuristics and knowledge. These will be presented in order of increasing use of imbedded knowledge. While it is true that these systems do not use the problem solving approach mentioned previously, it is because they are not attempting to learn at a child's level of innocence.

Winston's concept builder does not try to learn the philosophical meaning of "concept". Instead, it uses knowledge about the features of blocks world structures to build an internal representation of a concept. For example, imbedded knowledge and heuristics tell the system that an arch will have components with features such as AKO, supported by, and that some of these components must have a particular relationship while others are more flexible. The features are distinctly Winston's and thus the knowledge he gives the system causes it to build a concept as he

sees it. However, if different features: at right angles to, connected to, were chosen, a concept could still be built.

Rosenberg is too harsh on Hacker. While it is true that Hacker begins with enormous amounts of knowledge, it is not fair to say that it learns what a child learns by age two. The goal of Hacker is not merely to stack towers of blocks, but more importantly to write and debug programs which do so. When one is writing programs it is important to have an idea of the bugs one might encounter and to be able to draw on ones resources for programming techniques. These are not unrealistic skills for the system to possess before learning begins.

Systems such as Winston's and Sussman's use much knowledge and heuristics to learn, but further knowledge and heuristics are used by the teacher. Soloway's and Lenat's programs begin with the largest amount of imbedded knowledge because they need both a teacher's and a learner's knowledge.

The levels of knowledge and processing Soloway uses are advanced, perhaps even more advanced conceptually than the rules which are learned but this is not significant. Many newcomers to a sport know a good deal about sports and use that framework to understand the new sport. One first asks, "What's going on", that is, one focusses on action. Next one looks for competitive and cooperative activity and hypothesizes rules. All this requires

advanced, previously learned knowledge. Of course, if baseball were the first game one had seen, then the large amount of imbedded knowledge is inappropriate, but this is not seen as the goal of the program.

An argument can be made that the heuristics point too strongly to baseball and not competitive action sports in general. Consider basketball. It would be hard to focus on activity because there is always activity. The general back and forth movement across the court would have to be filtered out like the gum chewing of the pitcher or the warm up of a batter. The action of the game doesn't fit as nicely into episodes: would a basket or a foul shot be an episode? Where would the transfer of the ball or dribbling fit in?

AM has a huge knowledge base of 100 mathematical concepts and 250 heuristics which are chosen in such a way that the system can't help but discover mathematical concepts. Or put another way, AM discovers a great deal of interesting mathematical knowledge, but it discovers equally much uninteresting knowledge. Can we really call its work discoveries if it can't evaluate the worth of its creations? With the hindsight of a 20th century mathematician, Lenat was able to sort good math from bad, but if AM were working in a new field, and really discovering, not rediscovering, who would be able to sift through the mounds of results?

Lenat defends his heuristics by stating that when he noted a missing concept, he did not create a heuristic to make it appear.

> In that case, the new heuristic rule would simply be an encoding of a specific bit of mathematics which AM would then appear to discover using general methods. This must be avoided at all costs, even at the cost of intentionally giving up a certain discovery. If the needed rule is general - it has many applications and leads to many interesting results - then it really was an oversight not to include it in AM. [Lenat, 1976, p 154]

But what he is also saying is that in order to decide the original heuristics he thought about what was needed to produce concepts he deemed important and if a general concept with many applications was missed, he added another heuristic.

Another clue leads to the conclusion that AM's heuristics "give it all away". Lenat notes that the longer AM ran the further the concepts were from the original heuristics and that these concepts were too general and too weak to deal with notions such as primes. [Lenat, 1977, p839]

Lastly, I present an argument which is perhaps too anthropomorphic. At one point, Am's task is "If p+q=r, and p,q,r are primes. Then r is interesting if it has properties not possessed by p or by q." Lenat admits that such a search "showed a fundamental lack of understanding about numbers, addition, odd/even-ness, and primes." Can a system really discover without understanding? [Lenat, 1977, p839]

3. TEACHING SYSTEMS

3.1 The Teacher

In examining learning systems, I found that my focus of interest was captured by the teacher whose role in many systems was far more important than the learner. The teacher determined *what* was learned, *how* it was learned, and *whether* it was learned well enough. With the teacher having such powerful responsibilities, would it be possible to create a teaching system to automate the teacher's duties? In this chapter, I discuss the issues relating to this idea: How much knowledge does the teacher need? How is the training sequence chosen? How is evaluation handled? Where does the teacher get its knowledge (i.e., Who teaches the teacher?)? I will concentrate on the four example-driven systems: Winston's system, Hacker, Counting, and CEG. In the next section I will present a block diagram to model such systems.

3.1.1 Knowledge needed

The teacher need not be omniscient in order to teach effectively. Of course the teaching system needs to be aware of the same primitive knowledge and heuristics which are granted the learning system. The human trainer could choose his/her own set of primitives from the model world and drop them into the learning system's world model as givens. An easier choice to consider when creating a system is to assume that there exist global primitives which are already a part of both the teacher's and the learner's world model. This removes a difficult process from the teaching

system's domain and is not an unreasonable assumption. The teacher of Euclidean geometry doesn't need to choose the primitives because, as a 2000 year old subject, its primitives have been set for a long time.

A teaching system for Counting requires the least amount of domain knowledge. It needs to pose problems to the system and therefore must know the form to use in posing these problems, but it doesn't need to know the answer sequence of PCs and DCs. Selfridge found that human trainers spent too much of their time trying to program the system rather than provide it with a training sequince and so it could be an advantage for the teaching system not to be more knowledgeable.

The teacher for Hacker must be able to generate block tower instructions and to be able to order them for training purposes. Surprisingly, the trainer need not be competent in bug classification, patching, or programming since the Hacker system has that knowledge. The situation is similar to "management" stating a problem that needs to be solved and programmers doing it.

CEG must have the domain knowledge to pose legal problems, but if it randomly selects problems to be solved, not much learning will get done. For example, the ordered pairs may continually access the same line or may call upon retrieval examples without

exercising the modification routines.

Winston stands apart from the others because in his system the example posing and the learning have similar formats, meaning that the teacher's knowledge base must include the representation scheme. In Counting, the trainer provides a goal of pointer positions and counter numbers, and the system uses the PCs to achieve it. CEG, in the linear domain, is given ordered pairs and tries to use its examples-space and modifications to find and equation containing the two points. Hacker has a tower description as input and a program as output. But Winston provides an arch representation (albeit in pictoral form) and the system compare, and updates its present arch representation. The teaching system must have a complete representation of what is to be taught. If the trainer's knowledge is incomplete then either s/he is unaware of that fact and therefore passes on that incomplete knowledge believing it to be complete, or is aware of the lack and simply decides to teach what s/he knows. Then the learning system may generate an answer which doesn't fit the trainer's representation. In the latter case, the learning system has acted as teacher and the teacher's knowledge base is broadened. In a way, the knowledge of the teaching system mirrors that of the learning systems. While the learning system converts the line sketch of the arch into a semantic net, the teaching system must translate the network representation into a picture for the learner.

3.1.2 Choosing the training sequence

It is interesting that all these systems need the knowledge to pose problems and in most cases, a way of ordering those problems which will provide effective, efficient learning. Again this points out the importance of the training sequence. How will the teaching system generate that ordering, how will it know if it has taken steps too large or too small, and most of all, can the teaching system itself learn from mistakes and improve its performance?

Winston's is the most straightforward process to examine. The choice of the first example/near miss pair is important, after which examples and near misses can be put forward in any order. The teaching system has the complete representation of the concept. For an example, every object in the network must be in the drawing: a general term, such as "object" must be instantiated and that same instantiation used for the corresponding near miss. The "must" and "must not" directives must be followed for the example, and one of those is disregarded in the corresponding near miss. The example choosing continues in this manner. Examples differ from each other in the instantiations of terms, near misses differ in the choice of satellite pair or the addition of an extraneous object. One can see that what the system learns is the relationship of the satellite pairs and the hierarchy of generality.

In the other three system, the difficulty in choosing a training

sequence stems from the need to order the examples or problems from simple to complex. How can a teaching system judge complexity? Some parameters to consider are: 1. How long the example is. "Move the pointer all the way to the left" is more complex than "Move the pointer two spaces left". 2. How many different commands are in the example. "(AND (MAKE(ON A B))(MAKE (ON B C)))" is more complex than (MAKE(ON A B))". 3. How much this example differs from a previous one. "(AND(MAKE(ON A B))(AND(MAKE(ON B C))(MAKE(ON C D)))) loses its feeling of complexity when preceded by (AND(MAKE((ON A B))(MAKE (ON B C))).

None of these methods work for the linear domain of CEG where all the example problems are ordered pairs. A random selection of points will result in repetition of lines and uneven leaps in complexity. A teaching system for CEG would need to make use of the knowledge base of examples-space and modifications-space. It would use the learning techniques in reverse by creating instantiations of ordered pairs matching first the examples of examples-space, and then the modifications of those examples. This ordering would cause a progression of complexity. To create ordered pairs, it might use a fixed pair of Xs and solve the equations fo find the corresponding Ys. As shown earlier with Winston's work, there is a mirroring of the teaching and learning systems. The teaching system works from equation to ordered pair and the learning system from ordered pair to equation, with both going through examples-space and modifications-space.

In Counting, one can judge the complexity of an example in the training sequence by the three methods named. The problem here is how to choose the optimum number of subtasks or how large a step to take. Up to this point, I have assumed that a training sequence is determined then held fixed. This allows the teaching system no responsiveness to the behavior of the learner. A better scheme might be to choose two to three examples then evaluate how the learner is doing. Counting, with the black box style in which it is presently written, makes this nearly impossible. The teacher knows only whether the system solved the problem or not and how many trials before solution or time limit are reached. It would be far more useful to know details of which PCs and DCs were combined. Therefore, the teaching system will have to work blindly. Using the rules for judging complexity, it must create a sequence of subtasks. If unsuccessful, it must create another, taking larger or smaller steps but not knowing which is the proper direction. These iterations are continued until successful.

3.1.3 Evaluation

This process of altering the training sequence indicated the need for some sort of evaluator for the teaching system. The teacher should be able to judge the quality of the training sequence by the performance of the system and make changes accordingly. This is the teaching system's chance to learn.

Winston's program has the least flexibility as far as its capability for teacher learning. Each particular example or near

miss (except the first) teaches one specific aspect of the concept. This means that it doesn't matter where an example is introduced in the training system because the exact same information is learned. And since each example or near miss teaches exactly one fact, one can neither slow down nor speed up learning. On a matalevel, though, one can imagine the teacher improving the facets of the representation itself, for example, deciding that supported-by was not a useful relation and replacing it with something else. Winston must have done this himself as he built the system, but it would be difficult to automate this conceptual process.

In CEG and Hacker, the learning system evaluates its own progress. In Hacker the debugging mechanism checks how close the system is to solving the problem. In CEG, the judge counts how many constraints have been satisfied. Is there any reason for the teaching system to do evaluation or would it be redundant? There are two kinds of evaluation involved here, (1) evaluation of the success of the learning system, and (2) evaluation of the training sequence. Of course they are related. In order to evaluate the performance success of the learning system, the teaching system must share all the knowledge the learning system has. For example, in Hacker, the teaching system must have all the programming information the learning system has. In fact, it must have solved the problem itself and be able to compare its solution with that of the learner and judge whether differences between the two are significant. This does seem redundant to me. On the other hand, a more useful function of the teaching system's

evaluator is to judge the training sequence. Again, using Hacker as an example: the teaching system has made an estimate of the desired complexity for a problem. Its evaluator then monitors how long it takes the learner to solve the problem, that is, how many calls to the bug classifier are there, or how many patches are made. If there is a large difference between the time needed to solve this problem and the time used to solve its predecessor, then the training sequence is too complex and the teaching system can adjust its choice of the next problem accordingly. The evaluator may also run in different modes, sometimes making its judgements after every move, sometimes after every few moves, perhaps at times not at all.

The evaluating function may help the system recover in cases where the trainer makes an error. First, one must define what an error is. In Winston's system, an error would be calling an example a near miss or vice versa. Since all input is assumed error free, the system falls into a dilemna when faced with two different representations. If the learning system sent the teaching system evaluator the two conflicting answers, the evaluator could then monitor that node of the semantic net. It would keep a tally in future training examples and make a decision based on some threshold of "positiveness". In Hacker, an error would be an impossible tower. Here the evaluator, which is already judging the complexity of a problem, could simply have a cut off time, at which point the problem could be considered either too difficult, or impossible. Similarly, Counting's test limit (where it gives up after a certain number of iterations) serves the same purpose.

A time or iteration limit would be useful in CEG to prevent searching through examples-space and modifications-space for an answer that doesn't exist.

To me the appeal of creating a teaching system was that it would eliminate all the knowledge that the human trainer handed over to the learner. Instead the teaching system would handle that job. But, now, where does the teaching system get its knowledge? The human trainer has stepped back a step and is now a super teacher, teaching the teaching system. Could the super-teacher be automated, and then its super-teacher, and...?

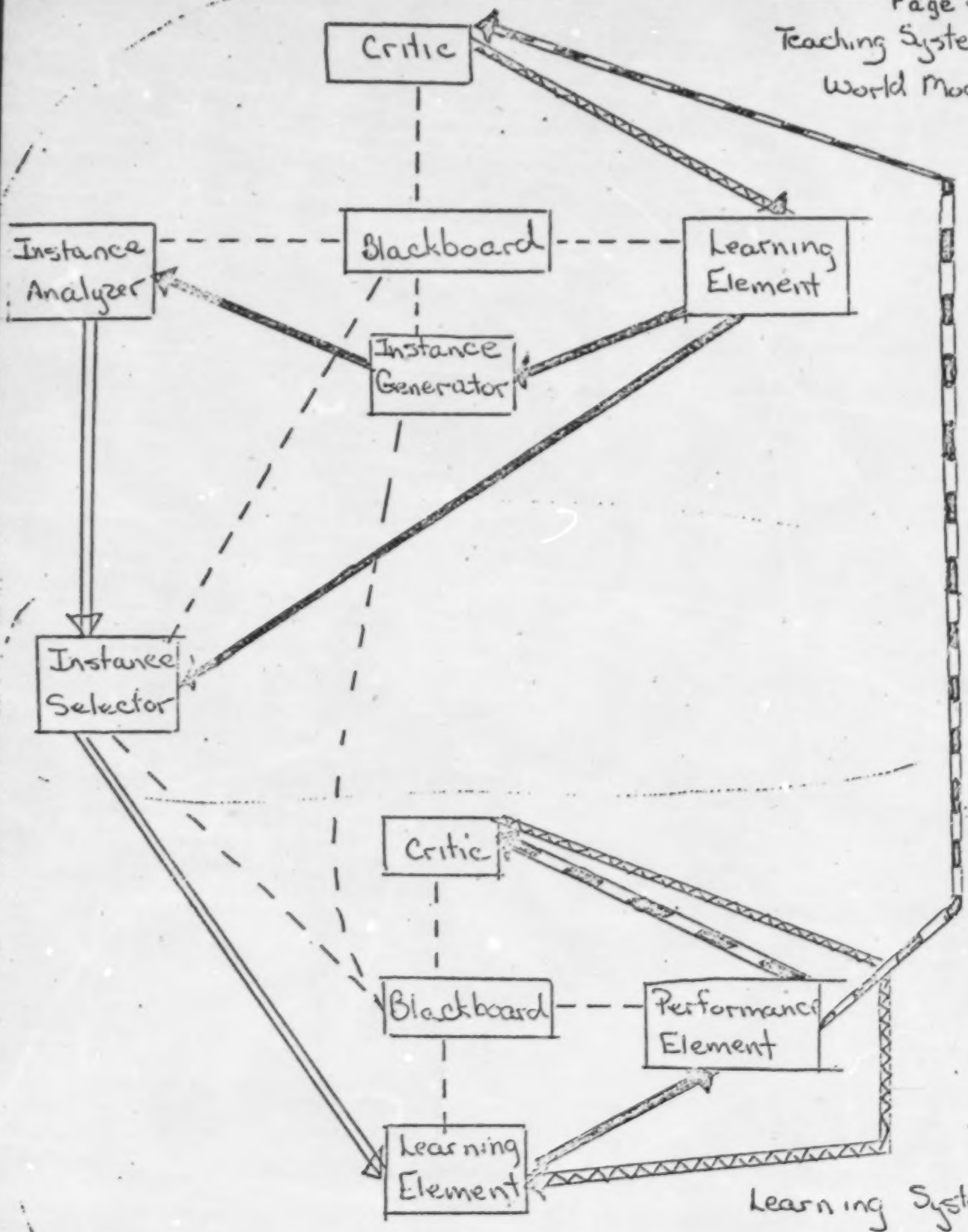
3.2 The Model

Just as there can be no General Problem Solver, it would be impossible to have a General Teaching System. In examining merely four systems, one can see te diversity in the role played by the teacher. It has been possible, however, to find common duties the teaching systems must perform even though they are done differently. To point out these like aspects, I have created a block diagram for the teaching system. Since a teaching system is relevant only in the context of a learning system, I have shown the interconnections between the two block diagrams. (The learning system diagram is Smith's.)

The primary role of the teaching system is generating good training problems. In that scheme, the instance generator uses information from its own knowledge base and that of the learning system to generate a stream of examples. These are ordered by the instance analyzer and one is chosen for presentation to the learner by the instance selector. The critic then uses information from the output (performance selector) of the learning system to diagnose problems in the training sequence and suggest improvement to the learning element. The learning element then makes suggestions for a new method of instance selection.

In both the learning system and the teaching system, the respective components communicate with each other through a blackboard (dashed black lines). The instance selector which is sometimes part of the teaching system and other times parts of the

Teaching System World Model



Learning System World Model

learning system must therefore communicate with both blackboards. The instance generator often receives some of its information from the knowledge base of the learning system. (Recall how I suggested that CEG use the modification routines to create examples.) and so it too must be connected to both blackboards.

Even though any system component has access to the information of any other, I have tried to clarify the primary channels of communication by marking them with solid lines. Note the symmetry of the functions of the two systems. In both cases, the learning element effects performance. In the learning system, newly learned information effects output by the performance element, whereas in the teaching system it causes changes in instance selection. The performance element needs to communicate with both critics: the learning system critic to evaluate errors and the teaching system to diagnose the cause of those errors in terms of the training sequence and suggest errors. Both systems in tern use the information from the critics to update the learning element.

3.2.1 World Model

The world model contains all of the fixed knowledge and heuristics in a system. The two world models overlap because they share some of the information in the knowledge base. The teaching system has access to all the information in the knowledge base of the learning system but because of the redundancy involved, usually chooses not to use it. In Smith's model he allowed the human

trainer to make changes to the world model of the learning system. Likewise the teaching system can make changes to the world model of the learning system but not to its own world model. Only a super-teacher which controlled the teaching system could do that.

3.2.2 Instance Group

The instance group is the heart of the teaching system since it creates and revises its training sequence. Initially it creates a group of examples and orders them. Selection is altered by feedback from the learning element.

The instance generator produces an unordered list of training examples. It generates this list in a variety of ways. Hacker uses its own world model to create block tower descriptions. Counting and CEG use the knowledge and heuristics of the learning system to build increasingly complex examples. Winston uses the representation net to produce instantiations of the concepts which are then translated into line drawings.

The instance analyzer takes the unordered list of examples and orders it from simple to complex. In general, it will use the rules for determining complexity mentioned previously. Sometimes, as in the case of CEG, the list is formed by creating increasingly complex examples and so is already ordered. At other times, as with Winston, such an ordering is unimportant. In these cases, the instance analyzer could be bypassed. One might also want the

instance analyzer to produce an alternate ordering such as group by aspect. For example, in Counting, to group all subtasks involving moving the pointer right.

The instance selector is shown between the learning system and teaching system because it can be handled by either system. The simplest algorithm it could use would be to select the examples serially beginning with the simplest. The learning element may later change this algorithm.

3.2.3 Critic

The critic observes the output of the performance element and evaluates the learning taking place. If learning is progressing smoothly, it may choose to intruct the learning element to speed things up. If the performance element is stumbling, spending long periods of time on a problem or finding problems impossible then the critic instructs the learning element to decrease the complexity of examples. The critic may also suggest that the instance analyzer use a different ordering, or, if an error from the trainer is suspected, the critic may suggest backtracking.

3.2.4 Learning Element

The learning element is the intermediary between the critic and the instance group. It takes the changes suggested by the critic and turns them into specific instructions for the instance generator. For example, when the critic says slow down, the

learning element decides how much. It records its own reactions to the critic's instructions and if it is changing its instructions too often, may choose to ignore the critic for a few rounds.

3.2.5 Summary This model is meant to show the integration of the teaching and learning systems. It also takes the process of preparing a training sequence, which is different for every system and isolates its components. It is hoped that if one wanted to build a teaching system, that this model would be a useful guide.

4. CONCLUSION

In this document, I have explored many aspects of the role of teacher and learner. In many areas of this study, interesting generalizations can be drawn. In other areas, the interest lies in the diversity of results.

The strongest theme throughout my study was that of "one step away". Those systems which were strongly directed needed a training sequence which provided careful steps so that examples did not differ from each other by such a large degree that the system could not draw generalizations. Those less strongly directed still needed the changes in environment to have the proper spacing which allowed them to adapt without wild fluctuations. Self directed systems used their own heuristics but needed to have tasks closely related or following a carefully monitored progression of complexity. Intuitively, I feel that people, too, learn incrementally. They are confused if the learning steps are too large. People do, on occasion make large leaps in learning (sometimes called the "aha" effect) when the incremental aspect of learning seems to have been completely abandoned. No system studied had any such facility.

There were many dual aspects in teaching/learning systems. By this I mean, as one characteristic of a system was increased, this necessitated or allowed the decrease of another characteristic.

One example which I have mentioned often is the tradeoff between imbedded knowledge and the power of the teacher. The more knowledge and heuristics the trainer took, the less was needed by the system. The systems which taught themselves, therefore needed the most knowledge. Another dual was the tradeoff between how structured the training was and how much evaluation was needed. In an error free, highly structured training sequence such as Winston's, no evaluation was needed. The input was reliable and the system knew how to handle it. When input was not as reliable, or the correct response was unknown, evaluation took a primary role, using a lot of comparison to judge the accuracy of a response. Most systems, of course, balanced their use of training and evaluation.

Initially, I was critical of systems which used a large amount of imbedded knowledge and heuristics. I came to believe that this was acceptable, in fact necessary when the systems were teaching themselves or when they were not attempting to be low level learners. In the former case the extra knowledge was acceptable because it was the same knowledge a trainer would use if the particular system had been teacher directed. In the latter case, I believe systems need not learn only from the novice stage.

A conclusion which surprised me was the relationship between the world models of the learning system and the teaching system. I had assumed that the learner's world model would necessarily be a subset of the teacher's, but it became apparent that they merely

overlapped. There was knowledge, such as types of bugs, that Sussman's system needed, that he as a teacher did not need in order to present tower examples.

Thus the teaching systems must have the knowledge and heuristics to create an incremental training sequence. The primitive knowledge of the learning systems reflects the level of knowledge to be learned. Either the teacher takes a dominant role or the learner has a strong evaluation process which takes the place of teacher criticism. Finally, all systems must wrestle with the conflict between the amount of knowledge a system is given and the power it displays.

5. APPENDIX

5.1 The Model

A Model For Learning Systems
by Smith, Mitchell, Cheslek, and Buchanan

Environment
Supervised-correct response available
Unsupervised-correct response not available. Needs to assign credit or blame for overall performance to individual responses.

Performance element-generates output. New learned information is communicated to it and affects its decisions.

Instance selector-chooses training sequence. Responsibility for selection can be external or internal and systems vary in sophistication (selecting instances to improve performance).

Critic-assesses behavior of performance element (evaluates), localizes reasons for poor performance (diagnoses) and suggests changes (therapist). The latter role is close to that of the learning element.

Learning element-translates abstract suggestions of critic to specific changes for performance element. Decides how much change should be made after each session.

Blackboard-broad k based information (e.g. rules) and temporary system oriented information (e.g. decisions made by performance element.

World Model-fixed conceptual framework within which system operates. May include domain specific heuristics. Change in world model produces layers of sophistication

5.2 Learning Systems

Counting
by O.G. Selfridge

1. Domain: Attaching a number to a group of objects

2. Purpose: Create a sequence of subtasks to teach system to count

3. Environment: Left and right moving pointer, number in a box which can be increased and decreased, objects on a table.

4. Performance element: Trainer combines primitive capabilities to form higher level derived capabilities.

5. Instance selector: The trainer forms higher and higher level DCs, noting that if the conceptual leap between problems is large, the problem is too hard for the system, and if the steps are too small, time is wasted learning what is already known.

6. Learning element: When trainer selects a goal such as set counter=0, the system systematically tries to achieve this goal using its primitives starting with a string (of PCs) of length one then 2... If in three tries, it is successful in achieving the goal, the goal becomes a DC.

7. Critic: Evaluator-When trainer thinks system is ready, s/he calls COUNT to combine PCs and DCs in a successful counting sequence. Diagnosis and therapy-If too many trial elapse without success, trainer is told to search for further subgoals.

8. World Model: PC1-move pointer left 1 position; Pc2- move pointer right 1 position; PC3-+1 to counter; PC4-subtract 1 from counter.

Constrained Example Generation
by E. L. Rissland and Elliot M. Soloway

1. Domain: Linear and Piecewise linear. To be extended to other domains.

2. Purpose: Attempt to solve problems using a primitive set of examples. If problem can't be solved using one of these primitive examples, use modification techniques.

3. Environment: Set of problems in form of constraints on lines and piecewise lines.

4. Performance Element: Decides equation of line or peicewise line which fits constraints.

5. Instance selector: Trainer chooses problem. Success increases with large number of modification methods and a careful selection of problems, whose solution will increase examples space.

6. Learning element: Compare current example to candidate examples. Judge records number of constraints met. If there is no perfect match modifier tries its procedures. If it, too, fails, the constructor attempts to build the example from scratch.

7. Critic: see learning element

8. World Model: Retrieval examples, modification routines and priority.

Learning Structural Descriptions From Examples [Smith, 1977]
by Patrick H. Winston

1. Domain Blocks world

2. Purpose: Learn to identify blocks world structures (such as arches and towers).

3. Environment: Set of possible line drawing/structure-classification pairs.

4. Performance Element: Decides class of structures to which the input structure belongs. Uses a model of the structure class supplied by the learning element.

5. Instance Selector: Accepts training instances supplied individually by the user.

6. Critic: Evaluation - compares the classification made by the Performance Element against the correct classification as supplied with each training instance. Diagnosis - generates a comparison description pointing out differences between the model and the structure description.

7. Learning Element: Constructs a model of the class of structures under consideration. Examines the comparison description supplied by the critic, and modifies the model to strengthen or weaken the correspondence between the model and the training instance.

8. World Model: Representation of scenes as line drawings, method of translating line drawings to graphical descriptions, grammar for representing the learned information, domain-specific heuristics for resolving among possible changes to each structure class model.

HACKER
by J. Sussman

1. Domain: Blocks world

2. Purpose: Given a tower building problem, learn to write a program to solve the problem.

3. Environment: Set of tower building problems. Noisefree, i.e. data always correct and problem always solvable. Unsupervised program, i.e. chooses its own method and evaluates self.

4. Performance Element: Uses learning element to write programs to solve specific problems.

5. Instruction selector: Accepts training instances supplied individually by user. Careful selection is important for success.

6. Critic: Evaluator-See if program gets stuck when 1) Primitive asked to do something impossible, 2) protection violation, or 3) aesthetic violation. If it does, HACKER must patch problem. Diagnosis-Six questions allow HACKER to classify bug. CAREFUL mode created a CHRONTEXT which identifies where bug occurred. Therapist- Constructs patch for bug.

7. Learning element-1)Look for program in Answer library. If it is there, instantiate it and check for bugs. If a bug exists, clasify it and patch it and repeat the test. 2)If not in answer library, use programming techniques to help propose a program. The critics gallery will help in testing for the efficiency of the algorithm. Again a search for bugs must be performed. 3) Any new program created (by patching and adding to a previous program) is put in the answer library.

8. World Model: Large set of primitives and heuristics: Answer library- primitive programs, such as PUTON; Blocks world knowledge; programming techniques-ideas for codeing which Sussman hopes are domain independent; Types of bugs-Sussman acknowledges that this is domain dependent and has devised six questions which classify all the possible bugs Hacker may encounter; Critics gallery-library of techniques for improving program efficiency.

Checker Player [Smith, 1977]
by A.L. Samuel

1. Domain: Game of Checkers

2. Purpose: Learn to play a good game of checkers by selection of a good set of board features.

3. Performance Element: Plausible moves.

4. Instance Selector: Entire set of possible training instances read from a list

5. Critic: Evaluation - analyses the learning ability of the performance element with the current set of evaluation function features. Diagnosis - singles out features which are not useful. Therapy - selects new features from a predefined list to replace useless features.

6. Learning Element: Redefines the current set of features as recommended by the critic.

7. World Model: Syntax of game board, form and features of linear polynomial evaluation function, method for adjusting evaluation function and rules of checkers.

Learning=Interpretation + Generalization
by E. Soloway

1. Domain: Action oriented competitive games

2. Purpose: Discovery of the rules of the game of baseball

3. Environment: Set of frame based snapshots of events in the game of baseball

4. Performance element: Filter out irrelevant actions in input, link actions together by inferring causal relationships and attribute purposes to players by hypothesizing competitive and cooperative goals for their actions.

5. Instamce Selector: This program is knowledge directed. Program is given a set of snapshots representing low level events in the game of baseball.

6. Learning Element: In order to discover general rules describing players' actions and goals, the input pattern descriptions need to be transformed. These initial snapshots pass through stages to activity relations to episodes to classes of goals of episodes. This is accomplished by levels of knowledge which cause the system to focus on activity, look for a plan structure..., and levels of processing which direct the system to focus and hypothesize and evaluate.

7. Critic: Evaluator-make predictions and judge reliability of predictions. Diagnosis and therapy-If prediction is shown to by false in an instance, its confidence value is decreased and vice versa.

8. World Model: Generate knowledge about types of goals and relationships found in competitive action games, common sense knowledge about pysical actions.

AM
by Douglas B. Lenat

1. Domain: Elementary Mathematics, mostly the area of sets.

2. Purpose: Discover interesting mathematical concepts

3. Environment: Elementary mathematics

4. Performance Element: Completes tasks to discover concepts or fill in facets of already discovered concepts.

5. Instance Selector: Uses a worth rating to decide what facet or concept to work on. Worth is increased if (1) a facet or concept was discussed recently, or (2) the concept is an extrema...

6. Critic: Evaluation - Worth rating tells the value of a discovery. AM just runs. No diagnosis or therapy takes place.

7. Learning Element: When a new concept is discovered, AM follows its agenda and heuristics to find the highest candidate for further study.

8. World model: 100 elementry mathematics concepts and 250 heuristics for evaluating worth.

Pandemonium: a Paradigm for learning
by O.G. Selfridge

1. Domain: Morse code letters

2. purpose: learn to distinguish morse code letters in noisy environment.

3. Environment: Set of possible morse code transmissions

4. Performance Element: four levels of demons recognize morse code dot or dash: first-accepts data, 2nd: compute-makes tests and assigns weights, 3rd:cognitive-computes weighted sum, 4th:decision-chooses most heavily weighted.

5. Instance selector-Training sequence provided by user

6. Critic:Evaluator-currently by user. Diagnosis-give answer and user says correct or incorrect.

7. learning element: Adaptive change occurs in two ways-hill climbing and subdemon selection (chg weights or throw out demon which does not contribute much and add a new one which is a composite of the best demons)

8. World model: Rules about length of dots and dashes; morse code alphabet.

The Simulation of Verbal Learning Behavior
by Edward A. Feigenbaum

1. Domain: three letter nonsense syllables
2. Purpose: learn to associate pairs of syllables
3. Environment: Set of 10-12 associated nonsense pairs
4. Performance Element: given stimulus, produces response syllable
5. Instance Selector: accepts syllable of author
6. Critic: evaluator-check net to see if discrimination is possible Trainer specifies if answer is correct. Diagnostician-It is always assumed that the error is caused by an incomplete net. Therapist- test more letters
7. Learning element: expand net by increasing number of letters examined
8. World Model: Visual means of distinguishing syllables. Building and traversing discrimination net.

5.3 Teaching Systems

Counting
by Oliver G. Selfridge

1. World Model: PCs, set of possible subtasks for counting, heuristics for enlarging or shrinking size of subtask.

2. Instance Group: Generator - systematic generation of all possible combination of PCs. Analyzer - Use complexity rules to choose simple to complex ordering. Also order by similar features. Selector - Use order from analyzer but doesn't use every example.

3. Critic: Tells if time limit is reached.

4. Learning Element: If many examples tried before time limit reached then probably sequence was in steps too small. If few examples tried before time limit reached then probably sequence was in steps too large.

CEG

by E. L. Rissland and Elliot M. Soloway

1. World Model: same as learning system

2. Instance Group: Generator - use examples-space and modification routines to create all possible problems. Analyzer - Eliminate duplicate problems. Selection - Start with retreival examples, then go to modification, then modification of modification.

3. Critic: Tells if one particular example or modification routine is overused or if problem is too hard.

4. Learning Element: Modifies selection of examples based on critic's feedback.

Learning Structural Descriptions from Examples
by Patrick H. Winston

1. World Model: Full representation net of concept, hierarchical representation of facets, all knowledge of learning system.

2. Instance Group: Generator - For examples: draw a picture from representation net using instances of all general facets (e.g., wedge for object). For near misses: draw a picture from representation net using the negative of a satellite pair (e.g., not supported by when net says must be supported by). Analyzer - Choose most general pair Selector - Use pair supplied by analyzer, than any example or near miss.

3. Critic: May find an example it hadn't considered

4. Learning Element: Modify net to allow for unforseen examples.

Hacker
by Gerald J. Sussman

1. World Model: Possible blocks world towers

2. Instance Group: Generator - Given a set of blocks, systematically finds all possible combinations of table arrangements and goal towers. Analyzer - Uses complexity rules to order problems. Selector - chooses an initial algorithm for choosing problems (e.g., every third problem from analyzer)

3. Critic: Feedback to Learning Element on success of problem solving.

4. Learning Element: Modify selection of problems.

6. BIBLIOGRAPHY

Boden, Margaret A. "Artificial Intelligence and Piagetian Theory." Synthese. 38:389-414, 1978.

Feigenbaum, Edward A. "The Simulation of Verbal Learning Behavior." Computers and Thought. (E.A. Feigenbaum and J. Feldman, Eds.) New York: McGraw Hill, 1963. pp71-105.

Lenat, Douglas B. A M : An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. PhD dissertation. Stanford:AI Lab, Stanford Computer Science Dept., 1976.

___. "Automated Theory Formation in Mathematics." Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Cambridge, 1977.

Rissland, E.L., and E.M. Soloway. "Overview of an Example Generation System." Proceedings first National Conference on Artificial Intelligence. Stanford, 1980.

Rosenberg, Jarett K. "Piaget and Artificial Intelligence". Proceedings of the First Annual National Conference on Artificial Intelligence. Stanford, 1980. pp266-268.

Samuel, A.L. "Some Studies in Machine Learning Using the Game of Checkers." Computers and Thought. (E.A. Feigenbaum and J. Feldman, Eds.) New York:McGraw Hill, 1963. pp71-105.

Selfridge, Oliver G. Learning to Count: How A Computer Might Do It. Cambridge:Bolt Beranek and Newman, Inc., 1979.

___. "Pandemonium: A Paradigm for Learning". Symposium on Mechanization of Thought Processes. London:Her Majesty's Stationery Office, 1959. pp513-525.

Smith, R.G., T.M. Mitchell, R.A. Chestek, and B.G. Buchanan. "A Model for Learning Systems". Proceedings Fifth International Joint Conference of Artificial Intelligence. 1977.

Soloway, Elliot M. "Learning+Interpretation:Generalization: A Case Study in Knowledge-Directed Learning. PhD dissertation. Amherst: COINS Technical Report 78-13, U of Mass., 1978.

Soloway, Elliot M. and Edward M. Riseman. "Levels of Pattern Description in Learning." Proceedings Fifth International Joint Conference of Artificial Intelligence, 1977. p801-811.

Sussman, Gerald Jay. A Computational Model of Skill Acquisition. PhD dissertation. Cambridge:AI Lab, MIT, 1973.

Winston, Patrick H. Learning Structural Descriptions from Examples. PhD dissertation. Cambridge:AI Lab, MIT, 1970.

___. "Learning Structural Descriptions from Examples". The Psychology of Computer Vision. (Winston, Ed.), New York:McGraw Hill, 1975.